Forschungszentrum Jülich

*Zentralinstitut für Angewandte Mathematik*

Interner Bericht

# Mathematik mit Maple
# Eine Einführung mit Beispielen aus der Analysis und Linearen Algebra

*Johannes Grotendorst*

FZJ-ZAM-IB-9820

**FORSCHUNGSZENTRUM JÜLICH GmbH**
**Zentralinstitut für Angewandte Mathematik**
**D-52425 Jülich, Tel. (02461) 61-6402**

Interner Bericht

# Mathematik mit Maple
# Eine Einführung mit Beispielen aus der Analysis und Linearen Algebra

*Johannes Grotendorst*

FZJ-ZAM-IB-9820

Dezember 1998

(letzte Änderung: 06.12.98)

# Inhaltsverzeichnis

# Vorwort

Das vorliegende Manuskript ist aus Kursen im Forschungszentrum Jülich und Lehrveranstaltungen an der Fachhochschule Aachen/Abteilung Jülich zum Thema "Einführung in Maple" bzw. "Mathematik mit Maple" entstanden. Nach einer Einführung in die Grundlagen des Computeralgebra-Systems Maple werden die vielfältigen mathematischen Funktionen des Systems an Beispielen aus der Analysis und Linearen Algebra demonstriert. Auf die Programmiermöglichkeiten in Maple wird ebenfalls eingegangen.

Die Maple-Kurse werden schon seit vielen Jahren in einem Ausbildungsraum, ausgestattet mit einem Pool von Rechnern (Workstations oder PCs) und einem Projektor für den Referenten, "interaktiv" durchgeführt. Dieses Skript und zusätzliches Übungsmaterial steht den Kursteilnehmern in Form von interaktiven mathematischen Dokumenten (Maple-Worksheets) zur Verfügung, die auf den Rechnern direkt ausgeführt und manipuliert werden können. Die Themengebiete sind so aufbereitet, daß sie auch leicht im Selbststudium bearbeitet und am Rechner nachvollzogen werden können. Die im Kurs behandelten Befehle und Funktionen sind in den zugehörigen Maple-Worksheets über Hyperlinks mit den entsprechenden Hilfeseiten des Maple-Systems verknüpft. Alle in diesem Skript verwendeten Beispiele wurden mit der Version Maple V Release 5 erstellt.

In vielen Bereichen der industriellen Entwicklung und der wissenschaftlichen Forschung ist die Anwendung von Computeralgebra-Systemen auf dem Vormarsch. Im Bereich der Ausbildung an Schulen und Hochschulen erleben wir derzeit die Einführung bzw. Erprobung dieser Mathematikwerkzeuge. Nach den Logarithmentafeln, Tabellenwerken für mathematische Funktionen, Integral- und Integraltransformationstafeln, Rechenschiebern und Taschenrechnern werden nun Computeralgebra-Systeme zum Standardwerkzeug von Naturwissenschaftlern, Ingenieuren und Technikern.

Die nächste Generation mathematischer Werkzeuge steht schon vor der Tür. Mit der Einführung standardisierter Protokolle (OpenMath, MathML) für den Austausch mathematischer Datenstrukturen wird eine effiziente Integration bzw. Kopplung von mathematischer Software, inbesondere von Computeralgebra- und Numerik-Software, angestrebt. So wird in der nächsten Maple-Version die einfache Nutzung von NAG-Programmen möglich sein.

Bei aller Begeisterung über das enorme Potential dieser Mathematikwerkzeuge darf der kritische Verstand bei der Anwendung dieser Systeme nicht abgeschaltet werden. Es sollte immer überlegt werden, ob die erhaltene Lösung sinnvoll ist (mathematisch oder physikalisch), ob die Größenordnung paßt oder was die Lösung in Spezialfällen besagt. Auch Mathematikprogramme machen Fehler!

Dieses LaTeX-Dokument wurde nahezu vollautomatisch aus dem zugrunde liegenden Maple-Dokument mittels der Interface-Funktion "Export As LaTeX" erzeugt. Lediglich der Index mit allen vorkommenden Maple-Kommandos wurde durch "Handarbeit" erstellt. Für die Durchführung dieser Arbeiten möchte ich Frau Monika Marx und Frau Daniela Beck danken.

Johannes Grotendorst, Dezember 1998

# Kapitel 1

# Grundlagen

## 1.1 Maple als Taschenrechner

```
> restart;
```

Maple kann wie ein konventioneller Taschenrechner für einfache arithmetische Operationen benutzt werden. Jede Eingabe muß durch ein Semikolon (;) oder einen Doppelpunkt (:) abgeschlossen werden. Bei Abschluß der Eingabe mit einem Semikolon wird die Ausgabe angezeigt, andernfalls unterbleibt die Anzeige des Ergebnisses. Einfache Operatoren sind **+, -, *, /, ^** bzw.** und **!**

```
> 1 + 2;
```

$$3$$

```
> 75 - 3;
```

$$72$$

```
> 5*3;
```

$$15$$

```
> 120/2;
```

$$60$$

Ganze und rationale Zahlen werden in Maple exakt verarbeitet, d.h. die Rechnungen erfolgen nach den Regeln der rationalen Arithmetik.

```
> 1/2 + 1/6;
```

$$\frac{2}{3}$$

Rationale Zahlen werden automatisch in der einfachsten Form ausgegeben, d.h. Zähler und Nenner haben keine gemeinsamen Teiler mehr. Maple kann mit beliebig großen Zahlen arbeiten:

```
> 100!;
```

$$933262154439441526816992388562667004907159682643816214685929638952175999932299156089414639761565182862536979208272237582511852109168640000000000000000000000$$

Diese Zahl hat

```
> length(%);
```

$$158$$

Ziffern und die folgende Primzahlzerlegung:

```
> ifactor(%%);
```

$$(2)^{97}\ (3)^{48}\ (5)^{24}\ (7)^{16}\ (11)^{9}\ (13)^{7}\ (17)^{5}\ (19)^{5}\ (23)^{4}\ (29)^{3}\ (31)^{3}\ (37)^{2}\ (41)^{2}\ (43)^{2}\ (47)^{2}\ (53)\ (59)\ (61)\ (67)\ (71)\ (73)\ (79)\ (83)\ (89)\ (97)$$

Auf die letzten drei Ergebnisse einer Rechnung kann mittels des Prozentzeichens % (**ditto**) zurückgegriffen werden. Dabei übergibt % das letzte, %% das vorletzte und %%% das vorvorletzte Ergebnis an die Kommandozeile. Der größte gemeinsame Teiler zweier natürlicher Zahlen:

```
> igcd(123, 45);
```

$$3$$

Klammern werden so verwendet, wie es in der Mathematik üblich ist.

```
> 2*(3+1/3) / (5/3-4/5);
```

$$\frac{100}{13}$$

Einige mathematische Konstanten sind in Maple bereits vordefiniert, z.B. bezeichnet

- Pi die Kreisteilungszahl und

- gamma die Eulersche Konstante.

Eine Übersicht über die vordefinierten symbolischen Konstanten erhält man mit:

```
> constants;
```

$$false,\ \gamma,\ \infty,\ true,\ Catalan,\ FAIL,\ \pi$$

Maple kann Gleitkommazahlen mit beliebig vielen Stellen verarbeiten:

```
> Pi;
```

$$\pi$$

```
> evalf(Pi, 100);
```

$$3.14159265358979323846264338327950288419716939937510582097494459230781640628620899862803482534211706\backslash 8$$

**evalf** berechnet einen numerischen Wert (f = floating-point) mit der angegebenen Stellenzahl, hier 100. Bei der Eingabe ist immer zu unterscheiden, ob eine Zahl mit oder ohne Dezimalpunkt eingegeben wird.

```
> 1/3;
```

$$\frac{1}{3}$$

```
> 1/3.;
```

$$.3333333333$$

```
> 3*%% -3*%;
```

$$.1\,10^{-9}$$

Wird in einem Ausdruck eine Zahl mit Dezimalpunkt angegben, dann werden alle vorkommenden arithmetischen Operationen numerisch mit einer vereinbarten Genauigkeit, hier mit 10 Dezimalstellen, durchgeführt.

```
> 2*(3.+1/3) / (5/3-4/5);
```

$$7.692307691$$

Maple kennt alle elementaren mathematischen Funktionen

```
> sin(Pi/4);
```

$$\frac{1}{2}\sqrt{2}$$

```
> sqrt(2);
```

$$\sqrt{2}$$

```
> sqrt(2.);
```

$$1.414213562$$

```
> exp(1);
```

$$e$$

```
> exp(1.);
```

$$2.718281828$$

```
> ln(1);
```

$$0$$

Eine Übersicht über die in Maple implementierten mathematischen Funktionen erhält man mit **inifcns** .

## 1.2 Wertzuweisung

```
> restart:
```

Wertzuweisungen (**assignment**) erfolgen in Maple nach der allgemeinen Vorschrift <lhs> := <rhs>; Die rechte Seite der Zuweisung <rhs> wird von Maple ausgewertet und der linken Seite <lhs> zugewiesen. Die linke Seite kann hierbei ein Name (**name**), eine Folge von Namen, ein indizierter Name oder ein Funktionsaufruf sein. Auf der rechten Seite können nicht nur mathematische Ausdrücke (**expression**), sondern auch Gleichungen oder Funktionen stehen. Für die Namen (Bezeichner) gelten die folgenden Regeln:

- Der Bezeichner muß mit einem Buchstaben beginnen.
- Danach können Zahlen, Unterstriche (_) oder weitere Buchstaben folgen.
- Sonderzeichen, wie z.B. %, #, @ u.a. sind nicht zugelassen.
- Namen von Maple-Befehlen dürfen nicht verwendet werden.
- Maple unterscheidet bei Namen zwischen Groß- und Kleinschreibung, so ist z.B. Var1 von var1 verschieden.

```
> Kreis_Umfang := 2 * Pi * r;
```

$$Kreis\_Umfang := 2\,\pi\,r$$

```
> Kreis_Flaeche := F = Pi * r^2;
```

$$Kreis\_Flaeche := F = \pi\,r^2$$

Die Eingabe eines Namens liefert den aktuellen Wert der entsprechenden Variablen:

```
> Kreis_Umfang;
```

$$2\,\pi\,r$$

```
> Kreis_Flaeche;
```

$$F = \pi\,r^2$$

Mehrfachzuweisungen sind ebenfalls möglich:

```
> U,F := 2*Pi*r,Pi*r^2;
```

$$U,\ F := 2\,\pi\,r,\ \pi\,r^2$$

```
> U;F;
```

$$2\,\pi\,r$$
$$\pi\,r^2$$

Solange einem Namen noch kein Wert zugewiesen wurde, steht dieser Name für sich selbst.

```
> r;
```

$$r$$

Eine Übersicht über alle in Maple vordefinierten Namen erhält man mit **ininames** .

Vorbesetzte Namen können durch Wertzuweisung nicht überschrieben werden:

```
> Pi := 3.14;

Error, attempting to assign to `Pi` which is protected
```

Der Namensschutz kann mit der Funktion **unprotect** aufgehoben werden:

```
> unprotect(Pi);
> Pi := 3.14;
```

$$\pi := 3.14$$

```
> evalf(2*Pi);
```

$$6.28$$

```
> r := 10;
```

$$r := 10$$

```
> Kreis_Flaeche;
```

$$F = 314.00$$

Das Löschen einzelner Wertzuweisungen erfolgt durch

<name> := '<name>';

```
> r:= 'r';
```

$$r := r$$

```
> Kreis_Flaeche;
```

$$F = 3.14\,r^2$$

```
> Pi := 'Pi';
```

$$\pi := \pi$$

```
> Kreis_Flaeche;
```

$$F = \pi\, r^2$$

```
> evalf(Pi);
```

$$3.141592654$$

```
> protect(Pi);
> Pi := 3.14;
```

```
Error, attempting to assign to `Pi` which is protected
```

Soll für alle nachfolgenden Rechnungen mit einer fest eingestellten Zahl von Stellen gerechnet werden, so muß die globale Variable **Digits** neu definiert werden:

```
> Digits:=100;
```

$$Digits := 100$$

```
> a:=1/3.;
```

$$a := .33333333333333333333333333333333333333333333333333333333333333333333333333333333\ 33333333333333333333333333333333333333$$

```
> '3*a - 1' = 3*a - 1;
```

$$3\,a - 1 = -.1\,10^{-99}$$

```
> sin(a);
```

$$.32719469679615224417334408526762060606430140689375979159005627707\ 05763744817615233969107939075693610$$

Die Eingabe eines Maple-Kommandos mit vorangestelltem Fragezeichen aktiviert die entsprechende Hilfeseite des Kommandos:

```
> ?solve
```

Es folgen einige typische Eingabefehler:

```
> b = sin(x)*cos(y)/(x*y);
```

$$b = \frac{\sin(x)\cos(y)}{x\,y}$$

```
> b;
```

$$b$$

```
> rhs(%%);
```

$$\frac{\sin(x)\cos(y)}{x\,y}$$

Fehlt bei einer Zuweisung der Doppelpunkt, so betrachtet Maple die Zuweisung als Gleichung. Hier fehlt eine Klammer um -1:

```
> v := b^-1;
```

```
`-` unexpected
```

```
> v := b^(-1);
```

$$v := \frac{1}{b}$$

Es muß grundsätzlich das Multiplikationszeichen (*) mit angegeben werden:

```
> 2x;
```

```
missing operator or `;`
```

```
> 2*x;
```

$$2\,x$$

Hier fehlt bei **convert** das zweite Argument:

```
> convert(tan(x));
```

```
Error, wrong number (or type) of parameters in function convert
```

```
> convert(tan(x), sincos);
```

$$\frac{\sin(x)}{\cos(x)}$$

Siehe auch **conver,sincos**

## 1.3 Funktionen

```
> restart:
```

Für die Definition von Funktionen gibt es in Maple die Pfeil-Notation **->**:

```
> f := x-> (x^4-2*x^3+3*x)/(2*x^2-4*x);
```

$$f := x \to \frac{x^4 - 2\,x^3 + 3\,x}{2\,x^2 - 4\,x}$$

```
> f(5.5);
```

$$15.55357143$$

```
> f(y);
```

$$\frac{y^4 - 2\,y^3 + 3\,y}{2\,y^2 - 4\,y}$$

Die Berechnung der Ableitungsfunktion f' erfolgt mit Hilfe des Operators **D** (nicht mit **diff**).

```
> f_strich:=D(f);
```

$$f\_strich := x \to \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> f_strich(5.5);
```

$$5.377551021$$

diff erwartet als Argument einen Ausdruck, also z.B. den Funktionsterm f(x):

```
> fstx:=diff(f(x), x);
```

$$fstx := \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

Namen mit Sonderzeichen müssen in Maple in **backquotes** eingeschlossen werden:

```
> `f'` := D(f);
```

$$f' := x \to \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> `f'(5)`= `f'`(5);
```

$$f'(5) = \frac{29}{6}$$

Durch **unapply** kann aus einem Ausdruck eine Funktion erzeugt werden:

```
> fst:=unapply(fstx, x);
```

$$fst := x \to \frac{4\,x^3 - 6\,x^2 + 3}{2\,x^2 - 4\,x} - \frac{(x^4 - 2\,x^3 + 3\,x)\,(4\,x - 4)}{(2\,x^2 - 4\,x)^2}$$

```
> h:= f_strich - fst;
```

$$h := f\_strich - fst$$

```
> h(x);
```

$$0$$

Komposition von Funktionen mit @

```
> f:= x-> sin(x);
```

$$f := \sin$$

```
> g:= x-> x^ 2;
```

$$g := x \to x^2$$

```
> h:= g@f;
```

$$h := g@\sin$$

```
> h(x);
```

$$\sin(x)^2$$

```
> g(f(x));
```

$$\sin(x)^2$$

Man beachte den Unterschied:

h:= g@f liefert als Ergebnis eine Funktion, aber g(f(x)) liefert einen Ausdruck.

Mehrfach geschachtelte Funktionen:

```
> (f@@3)(x);
```

$$(\sin^{(3)})(x)$$

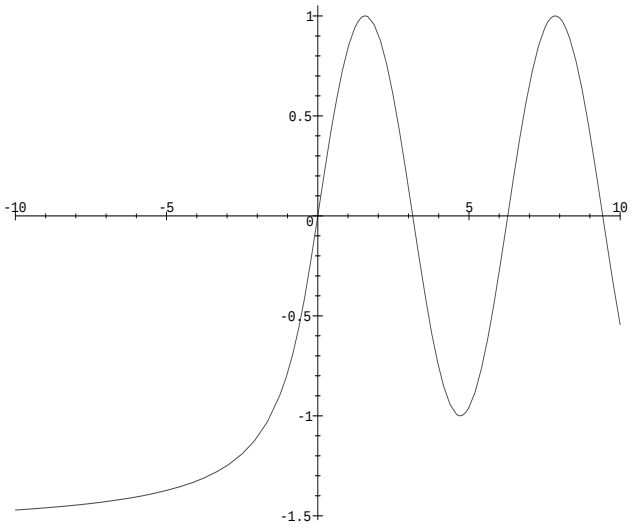
Stückweise definierte Funktionen:

```
> s := x -> if x < 0 then arctan(x) else sin(x) fi;
```

$$s := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ x < 0\ \mathbf{then}\ \arctan(x)\ \mathbf{else}\ \sin(x)\ \mathbf{fi}\ \mathbf{end}$$

Funktionen lassen sich in einfacher Weise graphisch darstellen:

```
> plot(s);
```




Berechnung einer Umkehrfunktion:

```
> y=exp(2*x);
```

$$y = e^{(2\,x)}$$

Gleichung nach x auflösen:

```
> solve(%, x);
```

$$\frac{1}{2}\ln(y)$$

```
> unapply(%, y);
```

$$y \to \frac{1}{2}\ln(y)$$

Das Volumen V einer Kugel ist eine Funktion des Radius r:

```
> V:= r-> 4/3* Pi*r^3;
```

$$V := r \to \frac{4}{3}\,\pi\,r^3$$

Der Radius hänge wiederum von der Zeit t ab (wie bei einem kugelförmigen Ballon, der mit Gas gefüllt wird). Um das Volumen V als Funktion von t darzustellen, nutzen wir wieder das Maple-Kommando unapply und erhalten so aus dem Ausdruck V(r(t)) eine Funktion der Zeit.

```
> V(r(t));
```

$$\frac{4}{3}\,\pi\,\mathrm{r}(t)^3$$

```
> V1 := unapply(V(r(t)), t);
```

$$V1 := t \to \frac{4}{3}\,\pi\,\mathrm{r}(t)^3$$

D(V1) liefert uns eine Funktion, die die Änderung des Volumens als Funktion der Zeit beschreibt.

```
> DV1 := D(V1);
```

$$DV1 := t \to 4\,\pi\,\mathrm{r}(t)^2\,\mathrm{D}(r)(t)$$

Definieren wir nun r als konkrete Zeitfunktion, dann erhalten wir bei Aufruf der Funktionen V1 und DV1 die zeitabhängigen Ausdrücke:

```
> r := t -> 1 - exp(-t);
```

$$r := t \to 1 - e^{(-t)}$$

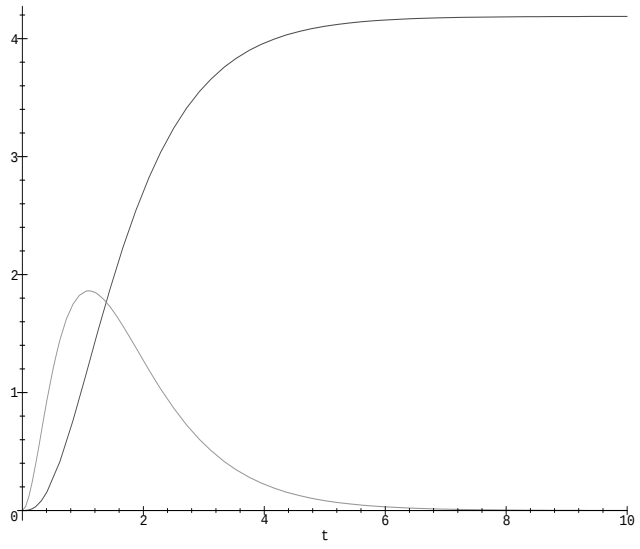
```
> V1(t);
```

$$\frac{4}{3}\,\pi\,(1 - e^{(-t)})^3$$

```
> DV1(t);
```

$$4\,\pi\,(1 - e^{(-t)})^2\,e^{(-t)}$$

```
> plot({V1(t), DV1(t)}, t=0..10);
```




## 1.4 Library-Struktur

```
> restart:
```

Ein Design-Kriterium bei der Entwicklung von Maple war, daß Maple auch auf Rechnern mit kleinem Hauptspeicher laufen sollte. Deshalb stehen nicht immer alle Funktionen direkt zur Verfügung (Maple V hat über 2700 Funktionen). Die Funktionen sind in verschiedene Bibliotheken und Pakete aufgeteilt:

- Standard Library
- Miscellaneous Library
- Funktionspakete für bestimmte Anwendungsgebiete
- Share Library

Die Funktionen der Standard Library stehen jederzeit zur Verfügung. Eine Übersicht erhält man mit **index,function**. Die Funktionen der Miscellaneous Library werden mit dem Kommando **readlib** geladen. Beispiel: Anwenden der Funktion **extrema**

```
> extrema( x*y, x^2+y^2=1, {x,y});
```

$$\mathrm{extrema}(x\,y,\, x^2 + y^2 = 1,\, \{x,\, y\})$$

```
> readlib(extrema):
```

Jetzt ist die Funktion extrema benutzbar:

```
> extrema(x*y, x^2+y^2=1, {x,y});
```

$$\{\frac{-1}{2}, \frac{1}{2}\}$$

Maple bietet für viele Anwendungsgebiete eigene Pakete. Eine Liste der verfügbaren Pakete erhält man mit **index,package**. Das Paket **combinat** enthält z. B. die Funktion fibonacci zur Berechnung der Fibonacci-Zahlen **combinat,fibonacci**. Die Funktion fibonacci kann nicht direkt aufgerufen werden:

```
> fibonacci(10);
```

$$\text{fibonacci}(10)$$

Beim Aufruf der Funktion ist zusätzlich noch der Name des Paketes anzugeben und zwar in der Form

<package_name>[<function >](<parameter>);

```
> combinat[fibonacci](10);
```

$$55$$

Mit dem Kommando **with** können die Funktionen eines Paketes geladen werden:

with(<package_name>, <function>);

lädt nur die Funktion <function>, mit

with(<package_name>);

werden alle Funktionen eines Paketes verfügbar gemacht.

```
> with(combinat, fibonacci);
```

$$[fibonacci]$$

```
> fibonacci(10);
```

$$55$$

Die anderen Funktionen des Paketes combinat stehen noch nicht zur Verfügung:

```
> permute([a, b, c], 2);
```

$$\text{permute}([a, b, c], 2)$$

```
> with(combinat);

Warning, new definition for Chi
```

$$[\text{Chi}, bell, \text{binomial}, cartprod, character, choose, composition, conjpart, decodepart, encodepart, fibonacci, firstpart, graycode, inttovec, lastpart, multinomial, nextpart, numbcomb, numbcomp, numbpart, numbperm, partition, permute, powerset, prevpart, randcomb, randpart, randperm, stirling1, stirling2, subsets, vectoint]$$

Jetzt können alle Funktionen des combinat Paketes direkt aufgerufen werden:

```
> permute([a, b, c], 2);
```

$$[[a, b], [a, c], [b, a], [b, c], [c, a], [c, b]]$$

```
> fibonacci(10);
```

$$55$$

Die Share Library (**share**) schließlich enthält Programme (Pakete, Funktionen) und Worksheets von Anwendern und einigen Maple-Entwicklern. Sie wird mit

```
> with(share);

See ?share and ?share,contents for information about the share library
```

$$[]$$

geladen. Der Zugriff auf einzelne Pakete oder Worksheets wird in einer ausführlichen alphabetischen Inhaltsübersicht (**share**,**contents**) beschrieben. Beispiel:

```
> with(trans);

Share Library:  trans

Author: Grotendorst, Johannes.

Description:  routines for rational function approximations to
rational points, Taylor and asymptotic series and functions.
```

$$[aitken,\ \varepsilon,\ gb,\ lev,\ ratgen,\ ratser,\ \rho,\ rhoit,\ rich,\ sidi,\ \theta,\ thetait]$$

Eine Inhaltsangabe sortiert nach Themengebieten (**share**,**index**,**subject**) und Autoren (**share**,**index**,**author**) ist ebenfalls vorhanden.

## 1.5 Einfache Datenstrukturen

```
> restart:
```

Es soll eine einfache Folge von Termen (expression sequence, **exprseq**) eingegeben werden:

```
> 1, 2, 3, 4, 5;
```

$$1,\ 2,\ 3,\ 4,\ 5$$

```
> expseq1 := 0, Pi/3, Pi/2, Pi;
```

$$expseq1 := 0,\ \frac{1}{3}\,\pi,\ \frac{1}{2}\,\pi,\ \pi$$

Eine Termfolge kann auch mit der Funktion **seq** erzeugt werden:

```
> expseq2 := seq(i^2, i=1..5);
```

$$expseq2 := 1,\ 4,\ 9,\ 16,\ 25$$

```
> expseq3 := seq (x^i, i=1..6);
```

$$expseq3 := x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6$$

Listen (**list**):

```
> data_list := [expseq2];
```

$$data\_list := [1,\ 4,\ 9,\ 16,\ 25]$$

```
> monomials := [expseq3];
```

$$monomials := [x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6]$$

Zugriff auf einzelne Elemente der Liste:

```
> data_list[3];
```

$$9$$

Anzahl der Listenelemente (**nops**):

```
> nops(monomials);
```

$$6$$

Liste in Termfolge verwandeln (**op**):

```
> op(monomials);
```

$$x,\ x^2,\ x^3,\ x^4,\ x^5,\ x^6$$

Mengen (**set**):

```
> set1 := {1,2,3,a,b,c};
```

$$set1 := \{a,\ b,\ c,\ 1,\ 2,\ 3\}$$

```
> set2 := {0,1,y,a};
```

$$set2 := \{a,\ y,\ 0,\ 1\}$$

Mengenoperationen:

```
> set1 union set2;
```

$$\{a,\ b,\ c,\ y,\ 0,\ 1,\ 2,\ 3\}$$

```
> set1 intersect set2;
```

$$\{a,\ 1\}$$

```
> set1 minus set2;
```

$$\{b,\, c,\, 2,\, 3\}$$

Anzahl der Elemente einer Menge:

```
> nops(set1);
```

$$6$$

Menge in Folge von Termen verwandeln:

```
> op(set1);
```

$$a,\, b,\, c,\, 1,\, 2,\, 3$$

Eine Funktion auf alle Elemente einer Menge (Liste) anwenden (**map**):

```
> map(sin, {expseq1});
```

$$\{0,\, 1,\, \frac{1}{2}\sqrt{3}\}$$

```
> map(sqrt, data_list);
```

$$[1,\, 2,\, 3,\, 4,\, 5]$$

Felder (**arrays**) sind mehrdimensionale Datenstrukturen. Die Feldelemente können direkt bei der Feldvereinbarung durch eine geschachtelte Liste spezifiziert werden

```
> A := array(-1..0,0..1,1..2,[[[1,x],[x,x^2]], [[1,y],[y,y^2]]]);
```

$$\begin{aligned} A := \mathrm{array}(&-1..0,\, 0..1,\, 1..2,\, [\\ &(-1,\, 0,\, 1) = 1\\ &(-1,\, 0,\, 2) = x\\ &(-1,\, 1,\, 1) = x\\ &(-1,\, 1,\, 2) = x^2\\ &(0,\, 0,\, 1) = 1\\ &(0,\, 0,\, 2) = y\\ &(0,\, 1,\, 1) = y\\ &(0,\, 1,\, 2) = y^2\\ &]) \end{aligned}$$

oder mittels Wertzuweisung auf die einzelnen Feldelemente definiert werden (als Indizes sind ganze Zahlen erlaubt):

```
> A[-1,1,2] := z^2;
```

$$A_{-1,\,1,\,2} := z^2$$

Matrizen (**matrices**) sind spezielle zweidimensionale Felder mit positiven Indizes:

```
> B := matrix(2,2,(i,j) -> x^(i+j-1));
```

$$B := \begin{bmatrix} x & x^2 \\ x^2 & x^3 \end{bmatrix}$$

Die Matrixelemente lassen sich direkt mit Hilfe einer anonymen Funktion generieren (dritter Parameter). Mit der Funktion map können alle Elemente einer Matrix oder einer Feldes manipuliert werden:

```
> map(diff,B,x);
```

$$\begin{bmatrix} 1 & 2\,x \\ 2\,x & 3\,x^2 \end{bmatrix}$$

Spezielle Felder für Hardware-Gleitkommazahlen (**hfarray**) erlauben die direkte Nutzung der CPU-Gleitkommaarithmetik.

```
> C := hfarray(1..2,1..2);
```

$$C := [[undefined,\, undefined],\, [undefined,\, undefined]]$$

```
> C[1,1] := Pi; C[2,1] := 2*Pi;
```

$$C_{1,\,1} := \pi$$

$$C_{2,\,1} := 2\,\pi$$

```
> print(C);
```

$$[[3.1415926535898, undefined], [6.2831853071796, undefined]]$$

Diese Gleitkommazahlen sind doppeltgenau (16 Stellen) nach Standard IEEE-754.

## 1.6 Strings, Namen

```
> restart:
```

In Maple werden drei Typen von Anführungszeichen (**quotes**) unterschieden:
- das Zeichen double-quote (") wird als Begrenzungszeichen für Zeichenketten (**strings**) verwendet;
- das Zeichen back-quote (‘) wird als Begrenzungszeichen für Namen (**symbols**) verwendet;
- das Zeichen single-quote (’) schließlich verhindert die Auswertung von Ausdrücken (**uneval**).

Aneinanderhängen von Strings (**cat**):

```
> str := "Dies ist" . " ein String";
```

$$str := \text{“Dies ist ein String”}$$

oder

```
> cat("Dies ist", " ein String");
```

$$\text{“Dies ist ein String”}$$

Länge des Strings (**length**):

```
> length(str);
```

$$19$$

Extrahieren und Suchen von Teilstrings (**substring**):

```
> substring(str, 3..3);
```

$$\text{“e”}$$

```
> substring(str, 3..8);
```

$$\text{“es ist”}$$

```
> searchtext("ist", str);
```

$$6$$

Maple-Ausdruck in String verwandeln (**convert,string**):

```
> convert(2/3, string);
```

$$\text{“2/3”}$$

```
> evalf(%);
```

$$\text{“2/3”}$$

String in Maple-Ausdruck zurückverwandeln (**parse**):

```
> parse(%);
```

$$\frac{2}{3}$$

```
> evalf(%);
```

$$.6666666667$$

Die bisher verwendeten Kommandos gelten auch für Namen (Symbole), z. B.

```
> ‘Dies ist‘ . ‘ ein Name‘;
```

$$Dies\ ist\ ein\ Name$$

Namen kann man Werte zuweisen:

```
> ‘Dies ist‘ . ‘ ein Name‘ := length(%);
```

$$Dies\ ist\ ein\ Name := 17$$

Strings können nicht auf der linken Seite einer Zuweisung stehen:

```
> "string" := 19;

Error, invalid lhs of assignment
```

Es folgen noch einige Beispiele, wie Namen (Symbole) in Maple verwendet werden können. Erzeugen von numerierten Namen:

```
> expseq := 1, 2, 3, 4, 5;
```

$$expseq := 1, 2, 3, 4, 5$$

```
> x.expseq;
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

oder direkt

```
> x.(1, 2, 3, 4, 5);
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

oder

```
> seq(x.k, k=1..5);
```

$$x1,\ x2,\ x3,\ x4,\ x5$$

Polynom mit unbestimmten Koeffizienten generieren mittels **add** :

```
> add(a.k*x^k, k=0..6);
```

$$a0 + a1\,x + a2\,x^2 + a3\,x^3 + a4\,x^4 + a5\,x^5 + a6\,x^6$$

Schöner wird dies mit indizierten Koeffizienten:

```
> add(a[k]*x^k, k=0..6);
```

$$a_0 + a_1\,x + a_2\,x^2 + a_3\,x^3 + a_4\,x^4 + a_5\,x^5 + a_6\,x^6$$

## 1.7 Ein- und Ausgabe über Dateien

```
> restart:
```

Mit dem **save** Kommando können die in Maple definierten Variablen in eine Datei geschrieben werden. Die Werte der Variablen werden dabei in linearer Ausgabeform (**lprint** Format) gespeichert. Diese Form hat den Vorteil, daß sie sich wieder als Eingabe für Maple-Befehle eignet. Es ist empfehlenswert, Dateinamen in Anführungszeichen (**doublequotes**) einzuschließen, so daß Maple sie als Strings interpretiert und nicht versehentlich als Ausdrücke behandelt. Dies ist insbesondere notwendig, wenn der Dateiname Sonderzeichen, wie z. B. einen Schrägstrich / (slash), einen Punkt . oder einen Bindestrich - enthält.

```
> a:= sin(x)/(x^2+1);
```

$$a := \frac{\sin(x)}{x^2+1}$$

```
> b:= diff(a,x);
```

$$b := \frac{\cos(x)}{x^2+1} - 2\,\frac{\sin(x)\,x}{(x^2+1)^2}$$

```
> save a,b, "datei.txt";
```

Werden die Variablen in eine Textdatei geschrieben, so müssen die Namen der Variablen beim save Kommando explizit als Parameter angegeben werden. Endet der Dateiname mit `.m`, so werden die Daten in einem internen binären Maple-Format gespeichert. Mit

```
> save "datei.m";
```

werden alle gebundenen Variablen in die Datei `"datei.m"` geschrieben. Dateien mit Binärformat können von Maple schneller eingelesen werden als normale Textdateien. Durch die Kommandos **writeto** und **appendto** werden alle folgenden Eingaben und die zugehörigen Ausgaben (in 2-dimensionalem Textformat) in eine Datei geschrieben:

```
> writeto("file.txt");
> c:=a+b;
> writeto(terminal):
> c;
```

$$\frac{\sin(x)}{x^2+1} + \frac{\cos(x)}{x^2+1} - 2\,\frac{\sin(x)\,x}{(x^2+1)^2}$$

```
> appendto("file.txt");
> d:=a*b;
> writeto(terminal):
```

Mit dem **read** Kommando lassen sich in Maple Text- und Binärdateien wieder einlesen:

```
> read "datei.txt";
```

$$a := \frac{\sin(x)}{x^2+1}$$

$$b := \frac{\cos(x)}{x^2+1} - 2\,\frac{\sin(x)\,x}{(x^2+1)^2}$$

```
> read "datei.m";
```

Beim Einlesen von Binrädateien werden die Wertzuweisungen nicht am Bildschirm angezeigt.

Das **writedata** Kommando schreibt numerische Daten von einem Vektor, einer Matrix oder einer Liste von Listen in eine Textdatei.

```
> A := array( [[1.5,2.2,3.4],[2.7,3.4,5.6],[1.8,3.1,6.7]] );
```

$$A := \begin{bmatrix} 1.5 & 2.2 & 3.4 \\ 2.7 & 3.4 & 5.6 \\ 1.8 & 3.1 & 6.7 \end{bmatrix}$$

```
> writedata(terminal,A,float);

1.5         2.2         3.4

2.7         3.4         5.6

1.8         3.1         6.7
> writedata("matrix.txt",A,float);
```

Das Kommando **readdata** liest zeilenweise Zahlen aus einer Datei und liefert eine Liste von Listen.

```
> readdata("matrix.txt",float,3);
```

$$[[1.5, 2.2, 3.4], [2.7, 3.4, 5.6], [1.8, 3.1, 6.7]]$$

```
> readdata("matrix.txt",[float,float,integer]);
```

$$[[1.5, 2.2, 3], [2.7, 3.4, 5], [1.8, 3.1, 6]]$$

```
> convert(%,matrix);
```

$$\begin{bmatrix} 1.5 & 2.2 & 3 \\ 2.7 & 3.4 & 5 \\ 1.8 & 3.1 & 6 \end{bmatrix}$$

Für das formatierte Lesen und Schreiben von Daten stehen die von der Programmiersprache C her bekannten Kommandos **scanf** und **printf** zur Verfügung.

Die Funktionen **latex**, **C** und **fortran** ermöglichen die Transformation von Maple-Ausdrücken nach LaTeX, C und Fortran:

```
> latex(b);

{\frac {\cos(x)}{{x}^{2}+1}}-2\,{\frac {\sin(x)x}{\left ({x}^{2}+1

\right )^{2}}}
> readlib(C): C(b);

   t0 = cos(x)/(x*x+1.0)-2.0*sin(x)/pow(x*x+1.0,2.0)*x;
> fortran(b);

   t0 = cos(x)/(x**2+1)-2*sin(x)/(x**2+1)**2*x
```

Die Ausgabe in eine Datei erfolgt gemäß

```
> latex(b,"datei.tex");
> C(b,filename="datei.c");
> fortran(b,filename="datei.f");
```

Weitere Ausgabe- und Transformationsmöglichkeiten (z.B. nach PostScript, LaTeX oder HTML) bietet die graphische Arbeitsoberfläche von Maple (**worksheet interface**).

## 1.8 Einfache Plotfunktionen

```
> restart:
```

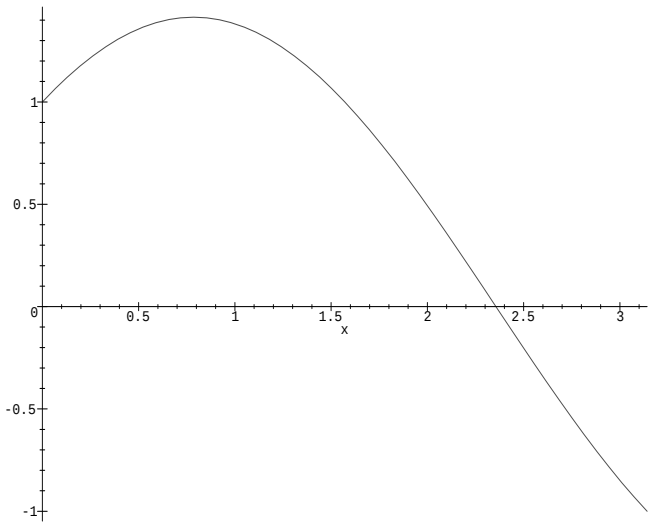
Zur graphischen Darstellung von mathematischen Formeln und Funktionen gibt es in Maple die Kommandos **plot** und **plot3d**. Weitergehende Plotfunktionen findet man in den Paketen **plots** und **plottools** bzw. in **DEtools,DEplot** und **stats,statplots**

Es folgt ein einfaches Beispiel zur graphischen Darstellung einer Funktion:

```
> f := x -> cos(x) + sin(x);
```

$$f := x \to \cos(x) + \sin(x)$$

```
> plot(f(x), x=0..Pi);
```




Die Graphik wird in das aktuelle Arbeitsblatt oder in ein separates Plotfenster gezeichnet, je nach Einstellung der Plot Display Option. Anschließend kann sie in der jeweiligen Plotumgebung über menügesteuerte Funktionen weiter verändert werden.

Das Plotten stückweise definierter Funktionen erfordert besondere Beachtung:

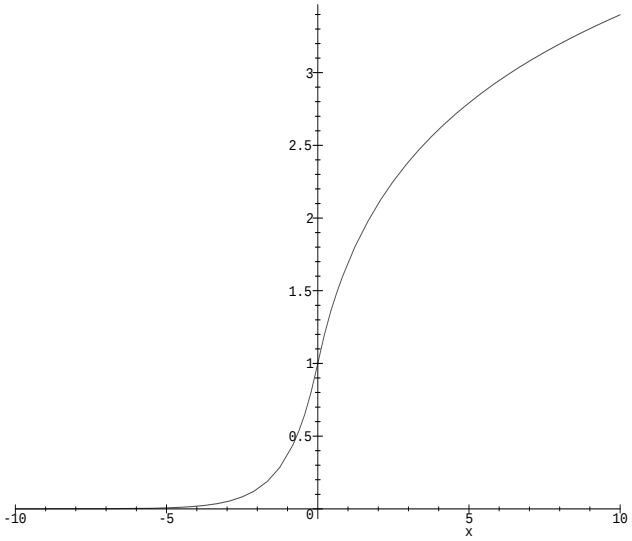
```
> s := x -> if x<0 then exp(x) else log(x+1) +1 fi;
```

$$s := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ x < 0\ \mathbf{then}\ \exp(x)\ \mathbf{else}\ \log(x+1) + 1\ \mathbf{fi}\ \mathbf{end}$$

```
> plot(s(x), x=-10..10);
Error, (in s) cannot evaluate boolean
```

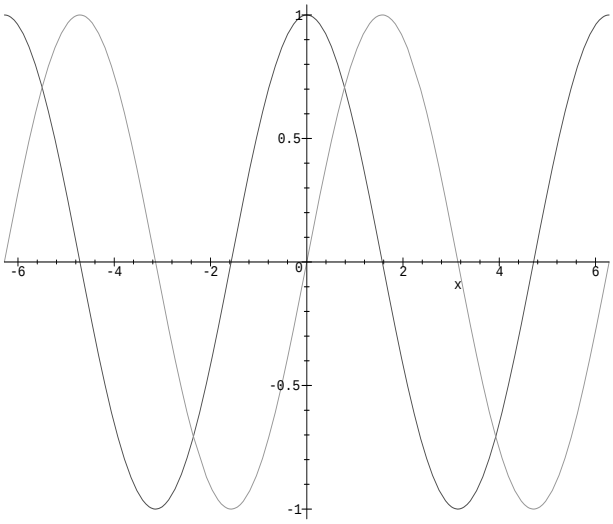
Beim Aufruf der Funktion plot wird zunächst der Funktionsausdruck s(x) symbolisch ausgewertet, d.h. Maple versucht den logischen Ausdruck x<0 auszuwerten, was zu der obigen Fehlermeldung führt. Dis symbolische Auswertung wird unterdrückt, wenn man die Eingabe s(x) in Anführungszeichen setzt:

```
> plot('s(x)', x=-10..10);
```



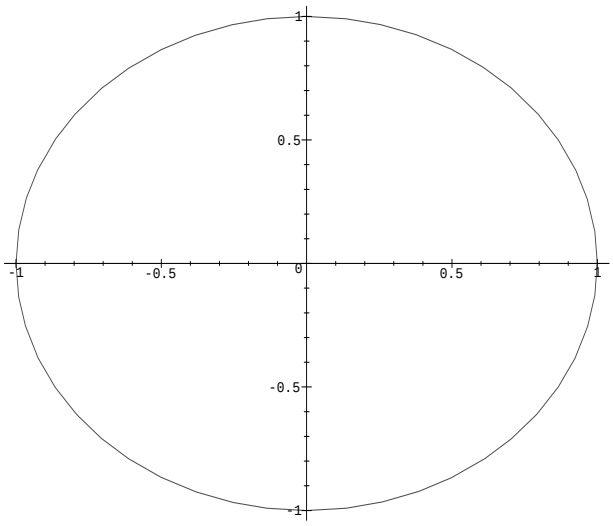

Darstellung von mehreren Funktionen in einem Schaubild:

```
> plot({sin(x), cos(x)}, x=-2*Pi..2*Pi);
```



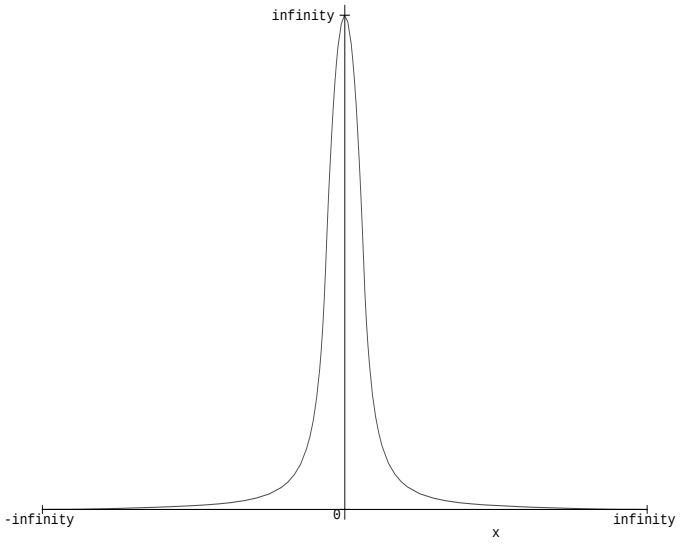

Darstellung einer Funktion, die in Parameterform gegeben ist:

```
> plot([sin(t), cos(t), t=-Pi..Pi]);
```




Soll der Funktionsverlauf für $x \to -\infty$ bzw. $x \to \infty$ dargestellt werden, so muß für die Plotbereichsgrenzen -infinity bzw. **infinity** eingesetzt werden.

```
> plot(1/x^2, x=-infinity..infinity);
```




Auch Punktmengen können graphisch dargestellt werden. Im nachfolgenden Beispiel werden die Koordinaten der Punkte

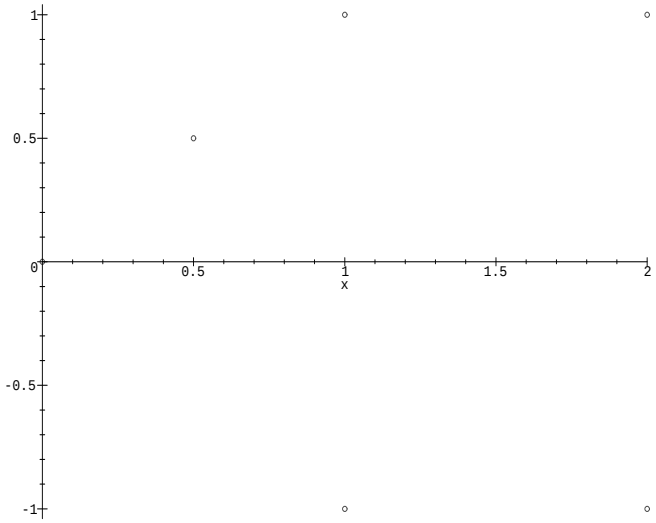
(0.5 , 0.5), (1 , 1), (2 , 1), (2 , -1), (1 , -1), (0 , 0)

als Liste an das plot Kommando übergeben:

```
> liste1 := [[0.5,0.5], [1,1], [2,1], [2,-1], [1,-1], [0,0]];
```

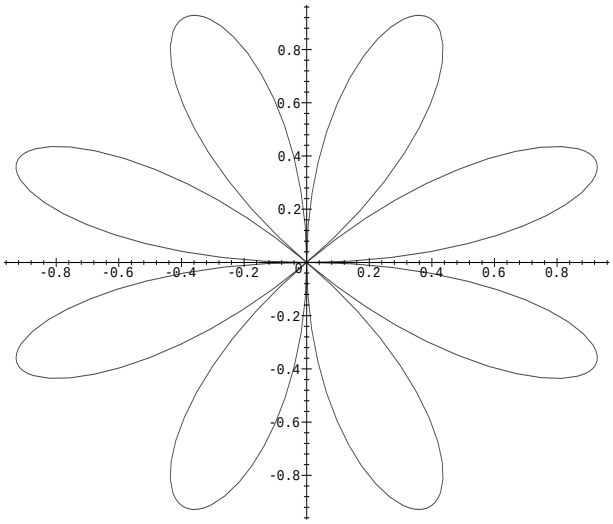
$$liste1 := [[.5,\ .5],\ [1,\ 1],\ [2,\ 1],\ [2,\ -1],\ [1,\ -1],\ [0,\ 0]]$$

```
> plot(liste1, x=0..2, style=point, symbol=circle);
```



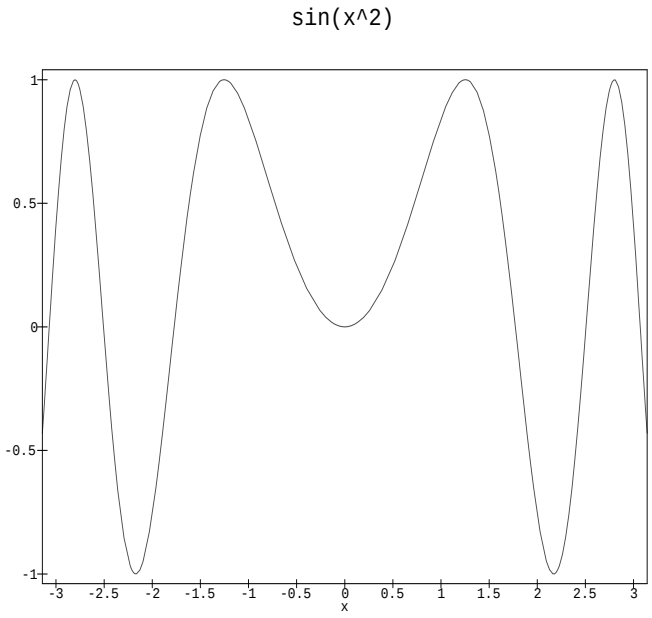

Darstellung mittels Polarkoordinaten:

```
> plot([sin(4*x), x, x=0..2*Pi], coords=polar);
```




Verschiedene Optionen können beim Aufruf des plot Kommandos gesetzt werden. Eine Liste der Optionen erhält man mit **plot,options** .

```
> plot(sin(x^2), x=-Pi..Pi, title="sin(x^2)", axes=boxed,
> xtickmarks=10, ytickmarks=5);
```



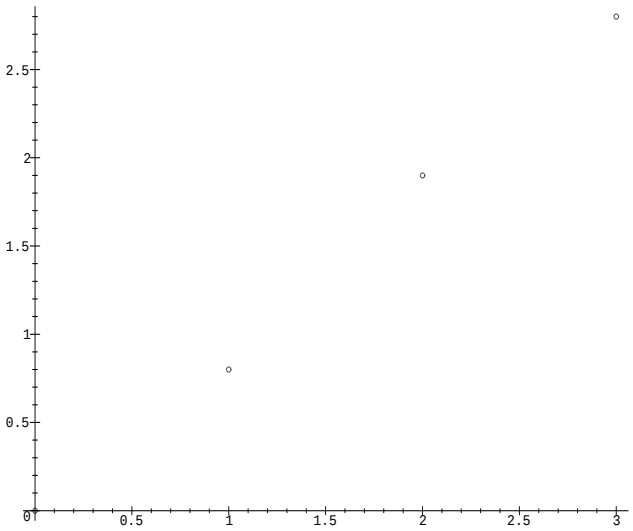

Die Datenstrukuren für 2D und 3D Plots lassen sich, wie andere Maple-Objekte auch, als benamte Objekte abspeichern. Mit dem display Kommando können die gespeicherten Graphiken dann visualisiert werden. So lassen sich z.B. Plots mit unterschiedlichen Attributen in einem Bild zusammenfassen.

Beispiel:

```
> liste2:= [[0,0], [1,0.8], [2,1.9], [3,2.8]];
```

$$liste2 := [[0, 0], [1, .8], [2, 1.9], [3, 2.8]]$$

```
> plt1 := plot(liste2, style=point, color=red, symbol=circle):
> plt1;
```




Wir wollen jetzt diese Daten sowie eine Ausgleichsgerade in einem Schaubild darstellen. Dazu werden zwei Graphiken mit unterschiedlichen Attributen erstellt. Die erste Graphik wird mit style=point erstellt und enthält nur die Datenpunkte. Danach wird mit der Funktion **leastsquare** aus dem Statistikpaket **stats,fit** die Ausgleichsgerade berechnet und diese Gerade mit dem Attribut style=line dargestellt.

```
> with(stats):
```

```
> with(fit):
```

```
> xl:= [seq(liste2[i][1], i=1..nops(liste2))];
```

$$xl := [0, 1, 2, 3]$$

```
> yl := [seq(liste2[i][2], i=1..nops(liste2))];
```

$$yl := [0, .8, 1.9, 2.8]$$

```
> leastsquare[[x,y], y=a*x+b, {a,b}]([xl, yl]);
```

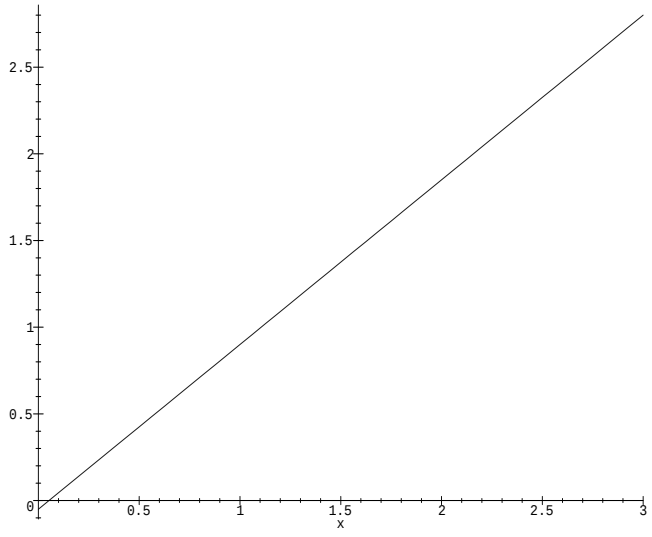
$$y = .9500000000\,x - .05000000000$$

```
> g := rhs(%);
```

$$g := .9500000000\,x - .05000000000$$

```
> plt2:=plot(g, x=0..3, style=line, color=blue):
```

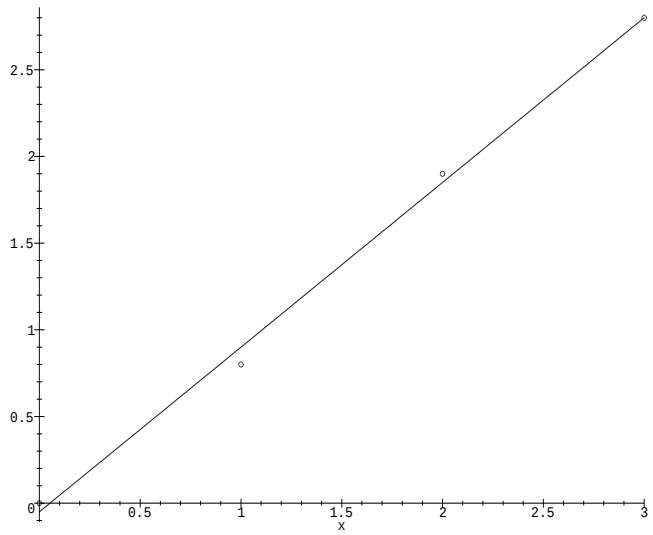
```
> plt2;
```




Beide Graphiken lassen sich nun mit dem Kommando display aus dem Paket plots in ein Bild zeichnen (**plots,display**):

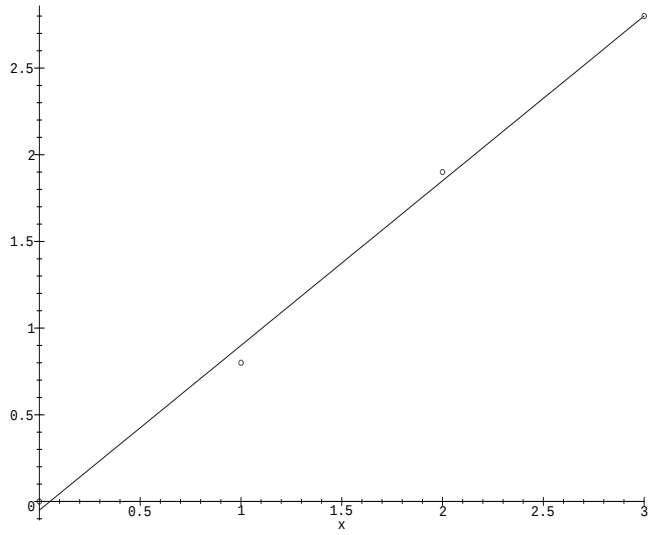
```
> with(plots):
```

```
> display({plt1, plt2});
```



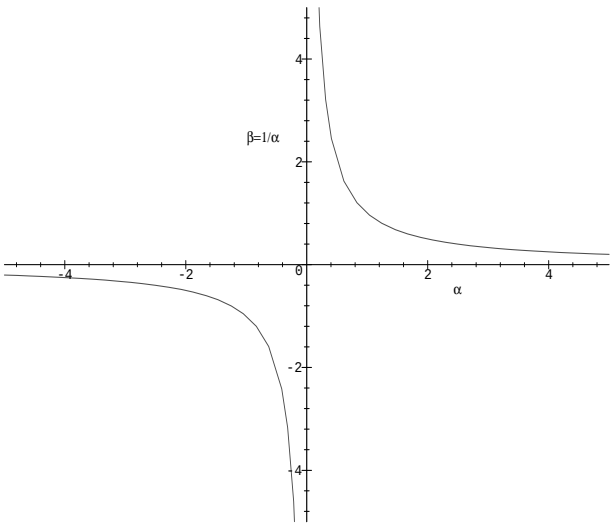

Mehrere 2D Graphiken mit unterschiedlichen Attributen können auch direkt mit dem Kommando plot veranschaulicht werden. Der folgende Befehl mit Optionslisten erzeugt die gleiche Graphik:

```
> plot([liste2, g], x=0..3, color=[red,blue],

> style=[point,line], symbol=[circle]);
```



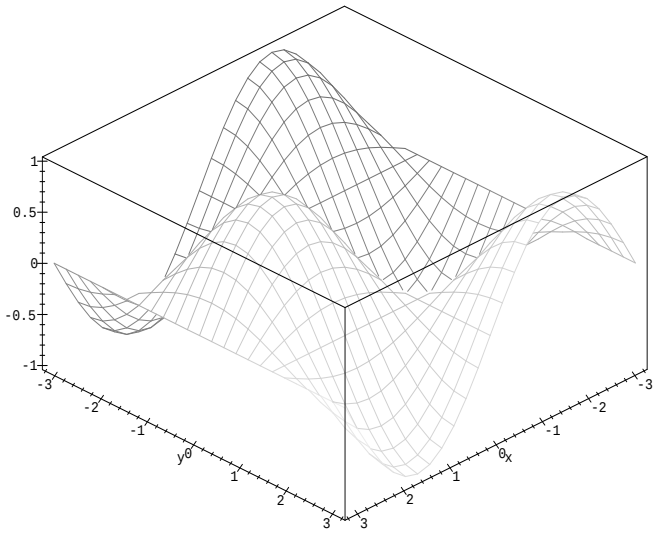

Eine Beschriftung der Koordinatenachsen mit griechischen Buchstaben erreicht man z. B. mit den folgenden Optionslisten:

```
> plot(1/a,a=-5..5,b=-5..5,labels=["a","b=1/a"],labelfont=[SYMBOL]);
```



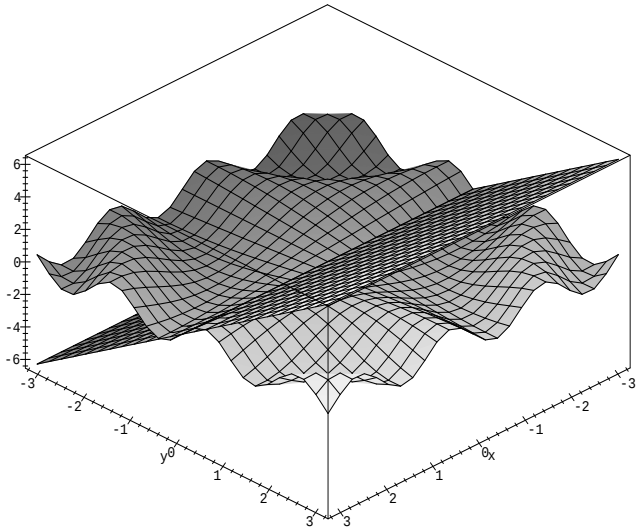

3D Graphiken werden mit dem Kommando **plot3d** erstellt. Es gibt hier ähnliche Optionen wie im 2D Fall.

```
> plot3d(sin(x)*cos(y), x=-Pi..Pi, y=-Pi..Pi, axes=boxed);
```



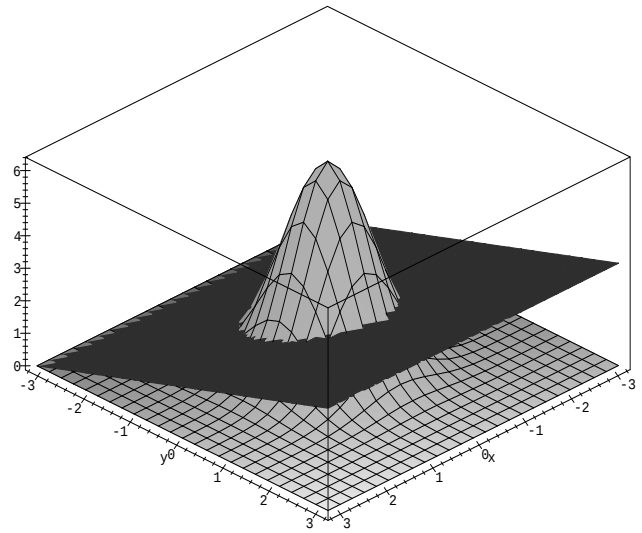
Es können auch mehrere Funktionen mit einem plot3d Aufruf veranschaulicht werden:

```
> plot3d({sin(x*y),2*y},x=-Pi..Pi,y=-Pi..Pi,style=patch,axes=boxed);
```



Falls 3D Graphiken mit unterschiedlichen Attributen in ein Bild sollen, kann man hierzu wieder die Funktion display verwenden:

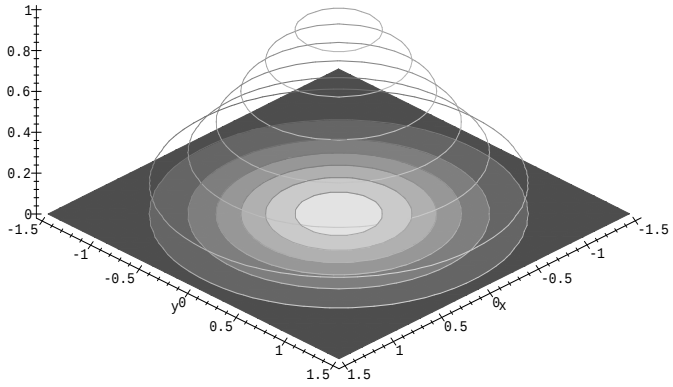
```
> plt1:=plot3d(2*Pi*exp(-x^2-y^2),x=-Pi..Pi,y=-Pi..Pi,style=patch):
> plt2:=plot3d(y/2+Pi/2,x=-Pi..Pi,y=-Pi..Pi,
> style=patchnogrid,color=navy):
> display({plt1,plt2},axes=boxed);
```



Die Funktion transform aus dem Plotpaket **plottools** ermöglicht die Transformation von Plotstrukturen. Eine Anwendung dieser Funktion ist z.B. die Einbettung eines 2D Plots in eine 3D Darstellung.

```
> p:=plot3d(exp(-x^2-y^2),x=-1.5..1.5,y=-1.5..1.5,style=contour,
> contours=[0.15, 0.3, 0.45, 0.6, 0.75, 0.9]):
> q:=contourplot(exp(-x^2-y^2),x=-1.5..1.5,y=-1.5..1.5,
> contours=[0.15, 0.3, 0.45, 0.6, 0.75, 0.9],filled=true):
```

```
> with(plottools):

> f := transform((x,y) -> [x,y,0]):
Warning, new definition for transform

> display({p,f(q)}, axes=frame);
```




Animationen lassen sich mit den Funktionen **animate** und **animate3d** aus dem Plotpaket plots erstellen.

```
> animate(sin(x*t),x=-10..10,t=1..2,frames=50,numpoints=100);

> animate3d(cos(t*x)*sin(t*y),x=-Pi..Pi,y=-Pi..Pi,t=1..2,style=patch);
```

Graphiken können in verschiedenen Formaten (**plot,device**) abgespeichert werden; mit der Funktion **plotsetup** wird die Graphikausgabe gesteuert. Die nachfolgende Graphik soll nun im PostScript-Format, ohne Bildrahmen und in Porträtdarstellung, in die Datei plotfile.ps geschrieben werden:

```
> plotsetup(ps,plotoutput="plotfile.ps",

> plotoptions="portrait,noborder");

> plot([t*cos(t),t*sin(t),t=0..5*Pi],title="Archimedische Spirale");
```

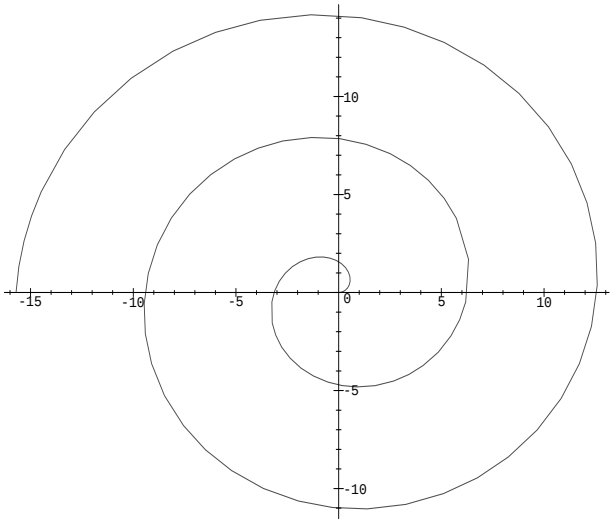
Anschließend werden wieder die Standardwerte für die Graphikausgabe in der aktuellen Benutzeroberfläche gesetzt:

```
> plotsetup(default);

> plotsetup();
```

$preplot = [], postplot = [], plotdevice = inline, plotoutput = terminal, plotoptions =$

```
> plot([t*cos(t),t*sin(t),t=0..5*Pi],title="Archimedische Spirale");
```

Archimedische Spirale



# Kapitel 2

# Manipulation von Ausdrücken und Datenstrukturen

## 2.1 simplify

```
> restart:
```

**simplify** - Anwendung von Vereinfachungsregeln auf Ausdrücke

```
> expr1 := 4^(1/2) + 3;
```

$$expr1 := \sqrt{4} + 3$$

```
> simplify(expr1);
```

$$5$$

```
> trig_expr := cos(x)^5+sin(x)^4+2*cos(x)^2-2*sin(x)^2-cos(2*x);
```

$$trig\_expr := \cos(x)^5 + \sin(x)^4 + 2\cos(x)^2 - 2\sin(x)^2 - \cos(2\,x)$$

```
> simplify(trig_expr);
```

$$\cos(x)^5 + \cos(x)^4$$

simplify kennt für viele Ausdrücke Vereinfachungsregeln. Wird eine bestimmte Regel als Argument spezifiziert, dann verwendet simplify nur diese Regel:

```
> expr2 := ln(x*y)+sin(x)^2+cos(x)^2;
```

$$expr2 := \ln(x\,y) + \sin(x)^2 + \cos(x)^2$$

```
> simplify(expr2, trig);
```

$$\ln(x\,y) + 1$$

```
> simplify(expr2, ln);
```

$$\ln(x\,y) + \sin(x)^2 + \cos(x)^2$$

Die Produktregel des Logarithmus wird erst angewendet, wenn die Argumente bestimmte Voraussetzungen erfüllen. Soll eine Vereinfachung ohne weitere Annahmen symbolisch durchgeführt werden, so kann man dies mit Hilfe des optionalen Parameters symbolic erreichen:

```
> simplify(expr2, ln, symbolic);
```

$$\ln(x) + \ln(y) + \sin(x)^2 + \cos(x)^2$$

simplify mit Annahmen:

```
> expr3 := sqrt(x^2);
```

$$expr3 := \sqrt{x^2}$$

```
> simplify(expr3);
```

$$\mathrm{csgn}(x)\,x$$

```
> ?csgn
> simplify(expr3, assume=real);
```

$$\mathrm{signum}(x)\,x$$

```
> simplify(expr3, assume=positive);
```

$$x$$

```
> simplify(expr2, ln, symbolic);
```
$$\ln(x) + \ln(y) + \sin(x)^2 + \cos(x)^2$$

simplify kann unter Nebenbedingungen vereinfachen; es werden allerdings nur polynomiale Ausdrücke als Nebenbedingung berücksichtigt:

```
> expr4 := x*y*z+x*y+x*z+y*z;
```
$$expr4 := x\,y\,z + x\,y + x\,z + y\,z$$
```
> simplify(expr4, {x*z=1});
```
$$x\,y + y\,z + y + 1$$

Die Form der Vereinfachung kann man durch die Angabe einer Liste mit Variablen steuern:

```
> expr5 := x^3+y^3;
```
$$expr5 := x^3 + y^3$$
```
> siderel := x^2+y^2=1;
```
$$siderel := x^2 + y^2 = 1$$
```
> simplify(expr5, {siderel}, [x, y]);
```
$$y^3 - x\,y^2 + x$$
```
> simplify(expr5, {siderel}, [y, x]);
```
$$x^3 - y\,x^2 + y$$

## 2.2 expand, factor

```
> restart:
```

**expand** - distributive Darstellung von Ausdrücken

```
> expand((x+1)*(x+2));
```
$$x^2 + 3\,x + 2$$
```
> expand(sin(2*x));
```
$$2\sin(x)\cos(x)$$
```
> expand(exp(a+ln(b)));
```
$$e^a\,b$$

Für Teilausdrücke kann die distributive Auswertung unterbunden werden:

```
> expand((x+1)*(x+y));
```
$$x^2 + x\,y + x + y$$
```
> expand((x+1)*(x+y), x+1);
```
$$(x+1)\,x + (x+1)\,y$$
```
> restart:
```

**factor** - Faktorisierung von Polynomen

```
> factor(x^2-1);
```
$$(x-1)\,(x+1)$$
```
> factor(x^3+y^3);
```
$$(x+y)\,(x^2 - x\,y + y^2)$$

Als Eingabe sind auch rationale Ausdrücke erlaubt; es werden dann Zähler und Nenner separat faktorisiert und anschließend gemeinsame Faktoren eliminiert.

```
> rat_expr := (x^16-y^16)/(x^7-y^7);
```
$$rat\_expr := \frac{x^{16} - y^{16}}{x^7 - y^7}$$
```
> factor(rat_expr);
```
$$\frac{(x+y)\,(x^2+y^2)\,(x^4+y^4)\,(x^8+y^8)}{x^6 + y\,x^5 + y^2\,x^4 + y^3\,x^3 + y^4\,x^2 + y^5\,x + y^6}$$
```
> expand(numer(%)*(x-y))/expand(denom(%)*(x-y));
```
$$\frac{x^{16} - y^{16}}{x^7 - y^7}$$

Ohne zweites Argument erfolgt die Faktorisierung im Ring der Koeffizienten (ganze Zahlen, rationale Zahlen, ...). Faktorisierung über bestimmte algebraische Zahlkörper ist ebenfalls möglich:

```
> factor(x^2+1);
```

$$x^2+1$$

```
> factor(x^2+1, I);
```

$$-(x+I)\,(-x+I)$$

```
> factor(x^4-2, sqrt(2));
```

$$-(x^2+\sqrt{2})\,(-x^2+\sqrt{2})$$

## 2.3 normal, rationalize

```
> restart:
```

**normal** - in rationalen Ausdrücken mit ganzen Koeffizienten werden gemeinsame Teiler in Zähler- und Nennerpolynom eliminiert.

```
> expr1 := (x^2 - y^2)/(x-y)^3;
```

$$expr1 := \frac{x^2-y^2}{(x-y)^3}$$

```
> normal(expr1);
```

$$\frac{x+y}{(x-y)^2}$$

normal wirkt rekursiv auf die Komponenten eines Ausdrucks:

```
> expr2 := sin((x*(x+1)-x)/(x+2))^2+cos((x^3+x^2)/(-(x^2+3*x+2)))^2;
```

$$expr2 := \sin(\frac{x\,(x+1)-x}{x+2})^2+\cos(\frac{x^3+x^2}{-x^2-3\,x-2})^2$$

```
> normal(expr2);
```

$$\sin(\frac{x^2}{x+2})^2+\cos(\frac{x^2}{x+2})^2$$

normal kennt keine Vereinfachungsregeln:

```
> simplify(%);
```

$$1$$

Die Option expanded liefert die distributive Darstellung von Zähler und Nenner:

```
> expr3 := 1/x+x/(x+1);
```

$$expr3 := \frac{1}{x}+\frac{x}{x+1}$$

```
> normal(expr3);
```

$$\frac{x+1+x^2}{x\,(x+1)}$$

```
> normal(1/x+x/(x+1), expanded);
```

$$\frac{x+1+x^2}{x^2+x}$$

Da normal den Zähler immer in distributiver Form darstellt, ist die Anwendung von normal nicht immer sinnvoll!

```
> expr4 := (x^20-1)/(x-1);
```

$$expr4 := \frac{x^{20}-1}{x-1}$$

```
> normal(expr4);
```

$$x^{19}+x^{18}+x^{17}+x^{16}+x^{15}+x^{14}+x^{13}+x^{12}+x^{11}+x^{10}+x^9+x^8+x^7+x^6+x^5+x^4 \\ +x^3+x^2+x+1$$

Hier ist factor sinnvoller.

```
> factor(expr4);
```

$$(x+1)\,(x^2+1)\,(x^4+x^3+x^2+x+1)\,(x^4-x^3+x^2-x+1)\,(x^8-x^6+x^4-x^2+1)$$

**rationalize** - Rationalisieren von Ausdrücken (Quadratwurzeln werden im Nenner beseitigt)

```
> (x+y)/(x+sqrt(3));
```

$$\frac{x+y}{x+\sqrt{3}}$$

```
> rationalize(%);
```

$$-\frac{(x+y)\,(-x+\sqrt{3})}{x^2-3}$$

## 2.4 convert

```
> restart:
```

**convert** - Umformung von Ausdrücken in äquivalente Darstellungen

```
> convert(cot(x), sincos);
```

$$\frac{\cos(x)}{\sin(x)}$$

Partialbruchzerlegung:

```
> convert((x^5+1)/(x^4-x^2), parfrac, x);
```

$$x+\frac{1}{x-1}-\frac{1}{x^2}$$

Gleitkommazahl in eine rationale Zahl umwandeln:

```
> pi := evalf(Pi);
```

$$\pi := 3.141592654$$

```
> convert(pi, rational, exact);
```

$$\frac{1570796327}{500000000}$$

In Maple wird eine Gleitkommazahl intern als Sequenz von zwei ganzen Zahlen (Mantisse und Exponent) dargestellt. Mit dem Kommando **op** läßt sich dieses Zahlenpaar anzeigen:

```
> piseq := op(pi);
```

$$piseq := 3141592654,\ -9$$

```
> piseq[1]*10^piseq[2];
```

$$\frac{1570796327}{500000000}$$

Gleitkommazahl durch eine rationale Zahl annähern:

```
> pi10 := convert(pi, rational, 10);
```

$$\pi 10 := \frac{104348}{33215}$$

```
> evalf(%);
```

$$3.141592654$$

pi10 und Pi stimmen in 10 Dezimalstellen überein:

```
> evalf(Pi - pi10);
```

$$0$$

```
> pi3:= convert(pi, rational, 3);
```

$$\pi 3 := \frac{22}{7}$$

pi3 und Pi stimmen in 3 Dezimalstellen überein:

```
> evalf(Pi - pi3, 3) ;
```

$$0$$

Typänderung von Datenstrukturen oder Ausdrücken:

```
> new_list := convert(a+b+c, list);
```

$$new\_list := [a,\ b,\ c]$$

```
> convert(new_list, set);
```

$$\{a,\ b,\ c\}$$

```
> taylor_entw := series(sin(x), x);
```

$$taylor\_entw := x - \frac{1}{6}\,x^3 + \frac{1}{120}\,x^5 + \mathrm{O}(x^6)$$

```
> subs(x=2, taylor_entw);

Error, invalid substitution in series
> taylor_poly := convert(taylor_entw, polynom);
```

$$taylor\_poly := x - \frac{1}{6}\,x^3 + \frac{1}{120}\,x^5$$

```
> subs(x=2, taylor_poly);
```

$$\frac{14}{15}$$

## 2.5 collect, combine

```
> restart:
```

**collect** - Zusammenfassen von Koeffizienten mit gleichen Faktoren

```
> poly1 := x^2+2*y*x-3*y+y^2*x^2;
```

$$poly1 := x^2 + 2\,y\,x - 3\,y + y^2\,x^2$$

```
> collect(poly1, x);
```

$$(1 + y^2)\,x^2 + 2\,y\,x - 3\,y$$

```
> collect(poly1, y);
```

$$y^2\,x^2 + (2\,x - 3)\,y + x^2$$

```
> trig_expr:=sin(x)*cos(x)+sin(x)+y*sin(x);
```

$$trig\_expr := \sin(x)\cos(x) + \sin(x) + y\sin(x)$$

```
> collect(trig_expr, sin(x));
```

$$(\cos(x) + 1 + y)\sin(x)$$

```
> poly2 := z*x*y+2*x*y+z;
```

$$poly2 := z\,x\,y + 2\,y\,x + z$$

```
> collect(poly2, x*y);

Error, (in collect) cannot collect, y*x
```

Koeffizienten von Produkten werden nicht zusammengefaßt; die Einführung einer Zwischengröße hilft hier aber oft weiter:

```
> subs(x=xyprod/y, poly2);
```

$$z\,xyprod + 2\,xyprod + z$$

```
> collect(%, xyprod);
```

$$(z + 2)\,xyprod + z$$

```
> subs(xyprod=x*y, %);
```

$$(z + 2)\,y\,x + z$$

Sollen die Koeffizienten mehrerer Variablen zusammengefaßt werden, so kann man die rekursive Methode (default) oder die distributive Methode (distributed) anwenden:

```
> poly3 := x*y+z*x*y+y*x^2-z*y*x^2+x+z*x;
```

$$poly3 := y\,x + z\,x\,y + y\,x^2 - z\,y\,x^2 + x + z\,x$$

```
> collect(poly3, [x, y]);
```

$$(1 - z)\,y\,x^2 + ((1 + z)\,y + 1 + z)\,x$$

```
> collect(poly3, [y, x]);
```

$$((1 - z)\,x^2 + (1 + z)\,x)\,y + (1 + z)\,x$$

```
> collect(poly3, [x, y], distributed);
```

$$(1 + z)\,y\,x + (1 + z)\,x + (1 - z)\,y\,x^2$$

**combine** - Zusammenfassen von Termen in Summen, Produkten und Potenzen

```
> exp(x)^2*exp(y);
```

$$(e^x)^2\,e^y$$

```
> combine(%, exp);
```
$$e^{(2\,x+y)}$$

combine wirkt in einigen Fällen wie die Umkehrung von expand.

```
> expand(%);
```
$$(e^{x})^{2}\,e^{y}$$
```
> (x^a)^2;
```
$$(x^{a})^{2}$$
```
> combine(%, power);
```
$$x^{(2\,a)}$$
```
> expand(%);
```
$$(x^{a})^{2}$$
```
> assume(x>0);
> expr1 := (x+1)^(1/2)*(x+2)^(1/2);
```
$$expr1 := \sqrt{x\tilde{}+1}\,\sqrt{x\tilde{}+2}$$
```
> combine(expr1);
```
$$\sqrt{x\tilde{}^{2}+3\,x\tilde{}+2}$$
```
> factor(%);
```
$$\sqrt{(x\tilde{}+1)\,(x\tilde{}+2)}$$
```
> expr2 := exp(sin(a)*cos(b))*exp(cos(a)*sin(b));
```
$$expr2 := e^{(\sin(a)\cos(b))}\,e^{(\cos(a)\sin(b))}$$
```
> combine(expr2);
```
$$e^{\sin(a+b)}$$
```
> expand(%);
```
$$e^{(\sin(a)\cos(b))}\,e^{(\cos(a)\sin(b))}$$

Zusammenfassen von Termen mit Integralen und Grenzwerten:

```
> x := 'x':
> Int(x,x=a..b) - Int(x^2,x=a..b);
```
$$\int_{a}^{b} x\,dx-\int_{a}^{b} x^{2}\,dx$$
```
> combine(%);
```
$$\int_{a}^{b} x-x^{2}\,dx$$
```
> Limit(x, x=a) * Limit(x^2, x=a) + c;
```
$$(\lim_{x\to a} x)\,(\lim_{x\to a} x^{2})+c$$
```
> combine(%);
```
$$\lim_{x\to a} x^{3}+c$$

## 2.6 sort

```
> restart:
```

**sort** - sortiert eine Liste mit Zahlen in aufsteigender Reihenfolge und die Terme eine Polynoms in absteigender Ordnung

```
> sort([1, 3, 2, 5, 3, 6, 3, 6]);
```
$$[1,\,2,\,3,\,3,\,3,\,5,\,6,\,6]$$

Eine Liste mit Strings wird lexikographisch sortiert:

```
> sort([a, z, x, e, f, r, b]);
```
$$[a,\,b,\,e,\,f,\,r,\,x,\,z]$$

Sortierung mit gegebener Sortierprozedur:

```
> f:=(x, y) -> if length(x) < length(y) then true else false fi;
```
$f :=$ **proc**$(x,\,y)$ **option** $operator,\ arrow$; **if** $\mathrm{length}(x) < \mathrm{length}(y)$ **then** $true$ **else** $false$ **fi end**

```
> sort([maple, blue, car, a], f);
```
$$[a,\ car,\ blue,\ maple]$$

Gemischte Listen werden nicht sortiert:

```
> big_list := [1, d, 2, 5, a, c, b, 9];
```
$$big\_list := [1,\ d,\ 2,\ 5,\ a,\ c,\ b,\ 9]$$
```
> sort(big_list);
```
$$[a,\ b,\ d,\ c,\ 1,\ 2,\ 5,\ 9]$$

Selektiv könnte etwa so sortiert werden:

```
> list1:=[d,a,c,b];
```
$$list1 := [d,\ a,\ c,\ b]$$
```
> list2:=[1,2,5,9];
```
$$list2 := [1,\ 2,\ 5,\ 9]$$
```
> list1:=sort(list1); list2:=sort(list2);
```
$$list1 := [a,\ b,\ c,\ d]$$
$$list2 := [1,\ 2,\ 5,\ 9]$$
```
> sorted_list:=[op(list1), op(list2)];
```
$$sorted\_list := [a,\ b,\ c,\ d,\ 1,\ 2,\ 5,\ 9]$$

Polynome werden in absteigender Ordnung sortiert:

```
> poly:=x^2+3*x+x^3+1;
```
$$poly := x^2 + 3\,x + x^3 + 1$$
```
> sort(poly);
```
$$x^3 + x^2 + 3\,x + 1$$

sort ersetzt Polynom poly durch das sortierte Polynom:

```
> poly;
```
$$x^3 + x^2 + 3\,x + 1$$

Polynome mit mehreren Variablen werden nach der Potenzsumme sortiert, die Variablenordnung kann angegeben werden:

```
> sort(x^2*y^3+y^2*x^2+x^3, [x, y]);
```
$$x^2\,y^3 + x^2\,y^2 + x^3$$
```
> sort(x^2*y^3+y^2*x^2+x^3, [y, x]);
```
$$y^3\,x^2 + y^2\,x^2 + x^3$$

## 2.7 map, zip

```
> restart:
```

**map** - Anwendung einer Maple-Funktion auf alle Operanden eines Ausdrucks oder einer Datenstruktur

```
> data_list := [0, x, y, exp(x), 3, -4];
```
$$data\_list := [0,\ x,\ y,\ e^x,\ 3,\ -4]$$
```
> map(t->t^2, data_list);
```
$$[0,\ x^2,\ y^2,\ (e^x)^2,\ 9,\ 16]$$
```
> map(sin, y+cos(x)+Pi/2);
```
$$\sin(y) + \sin(\cos(x)) + 1$$
```
> eqnset := {x^2+x=2, x^2=4};
```
$$eqnset := \{x^2 + x = 2,\ x^2 = 4\}$$
```
> map(lhs-rhs, eqnset);
```
$$\{x^2 - 4,\ x^2 + x - 2\}$$
```
> map(factor, %);
```
$$\{(x-2)\,(x+2),\ (x+2)\,(x-1)\}$$
```
> fcn_list := [sin(x), ln(x), x^2];
```
$$fcn\_list := [\sin(x),\ \ln(x),\ x^2]$$

```
> map(diff, fcn_list, x);
```

$$[\cos(x),\ \frac{1}{x},\ 2\,x]$$

**zip** - Verknüpfen zweier Listen

```
> list1 := [1, 15, 8];
```

$$list1 := [1,\ 15,\ 8]$$

```
> list2 := [2,6,12];
```

$$list2 := [2,\ 6,\ 12]$$

```
> zip(gcd, list1, list2);
```

$$[1,\ 3,\ 4]$$

```
> zip((x, y) -> x+y, list1, list2);
```

$$[3,\ 21,\ 20]$$

## 2.8 Strukturelle Manipulationen

```
> restart:
```

Linke Seite (**lhs**) und rechte Seite (**rhs**) einer Gleichung:

```
> eqn1 := x^2=3;
```

$$eqn1 := x^2 = 3$$

```
> lhs(eqn1);
```

$$x^2$$

```
> rhs(eqn1);
```

$$3$$

Extrahieren von Teilausdrücken mit **op**:

```
> poly := 1+x+x^2;
```

$$poly := 1 + x + x^2$$

```
> nops(1+x+x^2);
```

$$3$$

```
> op(poly);
```

$$1,\ x,\ x^2$$

```
> op(1, poly);
```

$$1$$

```
> op(3, poly);
```

$$x^2$$

```
> op(2, op(3, poly));
```

$$2$$

```
> op(2 .. 3, poly);
```

$$x,\ x^2$$

Substitution:

```
> subs(x=2, x^2+3*x-3);
```

$$7$$

**subs** führt syntaktische, keine algebraischen Substitutionen durch:

```
> subs(a+b=x, a+b+c);
```

$$a + b + c$$

Es werden nur die Operanden eines Ausdrucks ersetzt; a+b ist kein Operand des Ausdrucks a+b+c:

```
> op(a+b+c);
```

$$a,\ b,\ c$$

Für algebraische Substitutionen steht das allgemeinere Kommando **algsubs** zur Verfügung:

```
> algsubs(a+b=x, a+b+c);
```

$$x + c$$

subs führt keine Vereinfachungen oder Auswertungen durch, algsubs hingegen schon:

```
> subs(y=ln(x), exp(y));
```

$$e^{\ln(x)}$$

```
> eval(%);
```

$$x$$

```
> algsubs(y=ln(x), exp(y));
```

$$x$$

**subsop** ermöglicht die Substitution von Operanden in Ausdrücken:

```
> expr:=x^2;
```

$$expr := x^2$$

```
> subsop(2=3, expr);
```

$$x^3$$

## 2.9 Häufig gestellte Fragen

```
> restart:
```

Wie kann ich einen Wert für das Produkt von zwei Variablen substituieren? Durch Anwendung von simplify mit einer Nebenbedingung. Beispiel:

```
> expr1 := a^3*b^2;
```

$$expr1 := a^3\,b^2$$

```
> subs(a*b=5, expr1);
```

$$a^3\,b^2$$

```
> simplify(expr1, {a*b=5});
```

$$25\,a$$

Liefert simplify nicht die einfachste Form für einen Ausdruck, so erzielt simplify mit Nebenbedingung(en) oft das gewünschte Resultat.

```
> simplify(1-sin(x)^2);
```

$$\cos(x)^2$$

```
> simplify(1-cos(x)^2);
```

$$1-\cos(x)^2$$

```
> simplify(1-cos(x)^2, {1-cos(x)^2=sin(x)^2});
```

$$\sin(x)^2$$

Wie kann der konstante Faktor in den Ausdrücken 2x+2y oder -x-y ausgeklammert werden? Dies ist derzeit in Maple nicht möglich! Warum? Nun, weil für Maple Summen einfacher als Produkte sind (Design-Entscheidung in Maple: Konstante Zahlen als Faktoren werden automatisch auf alle Terme eines Ausdrucks verteilt). Für viele Fälle trifft dies auch zu, z. B.

```
> x^19 - x;
```

$$x^{19}-x$$

ist einfacher als

```
> factor(x^19-x);
```

$$x\,(x-1)\,(x^2+x+1)\,(x^6+x^3+1)\,(x+1)\,(1-x+x^2)\,(1-x^3+x^6)$$

-x-y ist für Maple somit einfacher als (-1)(x+y).

```
> -1*(x+y);
```

$$-x-y$$

Erzwungene Ausklammerung:

```
> expr2 := -x-y;
```

$$expr2 := -x-y$$

```
> subs(-1=vorz, expr2);
```

$$x\,vorz+y\,vorz$$

```
> factor(%);
```

$$vorz\,(x+y)$$

## 2.10 Fallstudie: Vollständige Induktion

```
> restart:
```

Wir setzen nun Maple zur Lösung eines mathematischen Problems ein, und zwar zum Beweis von Summenformeln (**sum**) mit vollständiger Induktion. Ist z.B. zu beweisen, daß

```
> Sum(f(k),k=1..m) = g(m);
```

$$\sum_{k=1}^{m} \mathrm{f}(k) = \mathrm{g}(m)$$

gilt, so muß man nur f und g definieren, der Induktionsbeweis wird dann weitgehend automatisch von Maple erledigt. Beispiel:

```
> f:=k->k/2^k;
```

$$f := k \to \frac{k}{2^k}$$

```
> g:=m->2-(m+2)/2^m;
```

$$g := m \to 2 - \frac{m+2}{2^m}$$

```
> Behauptung:=Sum(f(k),k=1 .. m) = g(m);
```

$$Behauptung := \sum_{k=1}^{m} \frac{k}{2^k} = 2 - \frac{m+2}{2^m}$$

Induktionsanfang: m = 1

```
> subs(m=1,Behauptung);
```

$$\sum_{k=1}^{1} \frac{k}{2^k} = \frac{1}{2}$$

```
> value (%);
```

$$\frac{1}{2} = \frac{1}{2}$$

Induktionsannahme: m = n

```
> Annahme:=subs(m=n,Behauptung);
```

$$Annahme := \sum_{k=1}^{n} \frac{k}{2^k} = 2 - \frac{n+2}{2^n}$$

Induktionsschluß: m = n+1

```
> Schluss:=Sum(f(k),k=1..n+1) = lhs(Annahme) + f(n+1);
```

$$Schluss := \sum_{k=1}^{n+1} \frac{k}{2^k} = \left(\sum_{k=1}^{n} \frac{k}{2^k}\right) + \frac{n+1}{2^{(n+1)}}$$

Mit der Annahme und geeigneter Formelmanipulation folgt nun

```
> lhs(Schluss) = rhs(Annahme) + f(n+1);
```

$$\sum_{k=1}^{n+1} \frac{k}{2^k} = 2 - \frac{n+2}{2^n} + \frac{n+1}{2^{(n+1)}}$$

```
> lhs(Schluss) = op(1,rhs(%)) +
simplify(normal(op(2,rhs(%))+op(3,rhs(%))));
```

$$\sum_{k=1}^{n+1} \frac{k}{2^k} = 2 - 2^{(-n-1)}\,(n+3)$$

qed.

Für Identitäten mit endlichen Produkten (**product**) läuft der Induktionsbeweis analog.

# Kapitel 3

# Grundlagen der Analysis

## 3.1 Folgen und Grenzwerte

```
> restart:
```

Eine Funktion f := IN -> M nennt man Folge mit Werten in M. Ist M=IR, so spricht man von einer reellen Zahlenfolge. Gegeben seien die Folgen

```
> a := n -> n^2/(2*n^2 +4*n - 1);
```

$$a := n \to \frac{n^2}{2\,n^2 + 4\,n - 1}$$

und

```
> b := n -> n*sin(1/n);
```

$$b := n \to n \sin(\frac{1}{n})$$

Endlich viele Folgenglieder können mit der Funktion **seq** berechnet werden.

```
> seq(a(n), n=1..10);
```

$$\frac{1}{5}, \frac{4}{15}, \frac{9}{29}, \frac{16}{47}, \frac{25}{69}, \frac{36}{95}, \frac{49}{125}, \frac{64}{159}, \frac{81}{197}, \frac{100}{239}$$

```
> seq(evalf(b(n)), n=5 ..9);
```

$$.9933466540, .9953767962, .9966021086, .9973978672, .9979436565$$

Die Grenzwerte dieser unendlichen Folgen lassen sich in Maple mit **limit** berechnen:

```
> Limit(a(n), n=infinity) = limit(a(n), n=infinity);
```

$$\lim_{n\to\infty} \frac{n^2}{2\,n^2 + 4\,n - 1} = \frac{1}{2}$$

```
> Limit(b(n), n=infinity) = limit(b(n), n=infinity);
```

$$\lim_{n\to\infty} n \sin(\frac{1}{n}) = 1$$

Die ersten zwanzig Glieder der Folge b werden zusammen mit ihrem jeweiligen Index als Liste mit 2-elementigen Listen abgespeichert, diese dient dann zur graphischen Anzeige der Folge:

```
> points := [seq([i, b(i)], i=1..20)]:
> p1 := plot(points, n=0..20, style=point):
```

In einer vorgegebenen $\varepsilon$-Umgebung

```
> epsilon := 10^(-3);
```

$$\varepsilon := \frac{1}{1000}$$

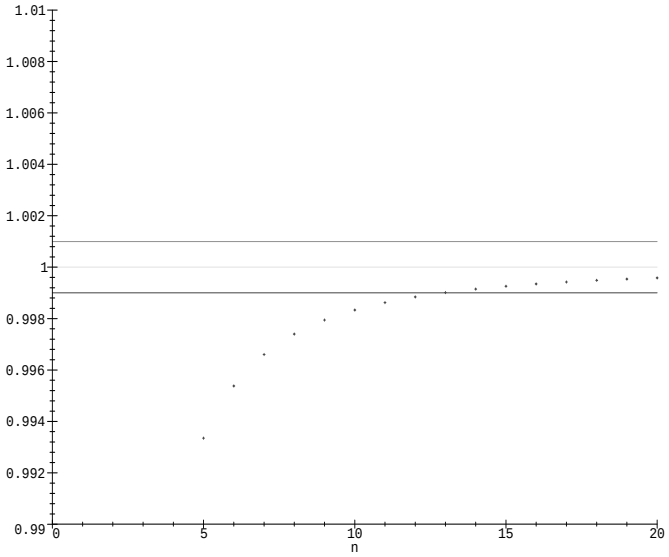
um den Grenzwert von b

```
> q := limit(b(n), n=infinity);
```

$$q := 1$$

befinden sich dann alle bis auf endlich viele Folgenglieder (Definition des Grenzwertes einer unendlichen Folge).

```
> p2:=plot({q-epsilon,q,q+epsilon},n=0..20,q-10*epsilon..q+10*epsilon):
> plots[display]({p1, p2});
```




Die nächste Folge hat keinen Grenzwert, dafür aber zwei Häufungspunkte.

```
> c := n -> (-1)^n*(n+2)/(n+cos(n));
```

$$c := n \to \frac{(-1)^n (n+2)}{n+\cos(n)}$$

Mehrere Häufungspunkte einer Folge werden von Maple durch den Bereichsoperator a..b (**range**) angezeigt.

```
> Limit(c(n), n=infinity) = limit(c(n), n=infinity);
```

$$\lim_{n\to\infty} \frac{(-1)^n (n+2)}{n+\cos(n)} = -1..1$$

Bei divergenten Folgen meldet Maple $\infty$ bzw. $-\infty$:

```
> d[1] := n -> 2^n;
```

$$d_1 := n \to 2^n$$

```
> Limit(d[1](n), n=infinity) = limit(d[1](n), n=infinity);
```

$$\lim_{n\to\infty} 2^n = \infty$$

```
> d[2] := n -> -2^n;
```

$$d_2 := n \to -2^n$$

```
> Limit(d[2](n), n=infinity) = limit(d[2](n), n=infinity);
```

$$\lim_{n\to\infty} -2^n = -\infty$$

## 3.2 Grenzwerte bei Funktionen, Stetigkeit

```
> restart:
```

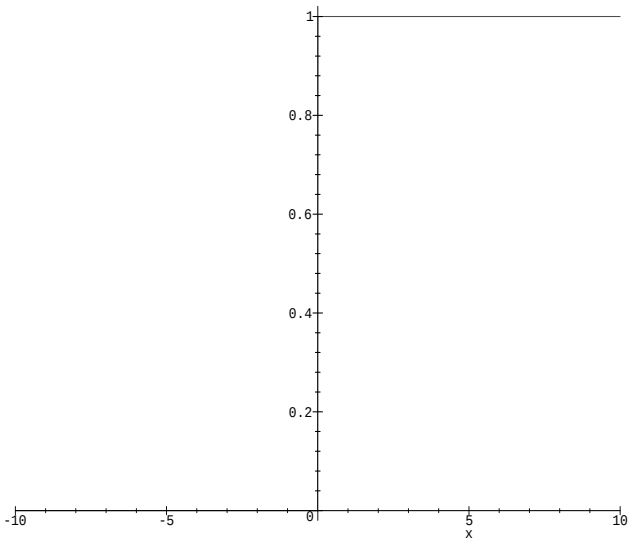
Wir betrachten die Stufenfunktion **Heaviside** und führen die Abkürzung

```
> alias(h=Heaviside);
```

$$I,\ h$$

ein. Mit dem **alias** Kommando kann der Benutzer die Maple-Notation seinen Bedürfnissen ensprechend anpassen. Die Heaviside-Funktion ist eine Stufenfunktion

```
> plot(h(x), x=-10..10);
```




die wir an einigen Stellen untersuchen wollen, d.h wir interessieren uns für Grenzwerte der Form

```
> Limit(h(x), x=a);
```

$$\lim_{x\to a} \mathrm{h}(x)$$

Für $x \to -\infty$ bzw. $x \to \infty$ erhalten wir

```
> Limit(h(x), x=-infinity): % = value(%);
```

$$\lim_{x\to(-\infty)} \mathrm{h}(x) = 0$$

```
> Limit(h(x), x=+infinity): % = value(%);
```

$$\lim_{x\to\infty} \mathrm{h}(x) = 1$$

An der Stelle x=0 befindet sich eine Unstetigkeitsstelle:

```
> Limit(h(x), x=0): % = value(%);
```

$$\lim_{x\to 0} \mathrm{h}(x) = \mathit{undefined}$$

Der linksseite Grenzwert

```
> Limit(h(x), x=0, left): % = value(%);
```

$$\lim_{x\to 0-} \mathrm{h}(x) = 0$$

und der rechtsseitige Grenzwert

```
> Limit(h(x), x=0, right): % = value(%);
```

$$\lim_{x\to 0+} \mathrm{h}(x) = 1$$

stimmen nicht überein. Die Stetigkeit einer Funktion kann auch mit der Prozedur **iscont** überprüft werden:

```
> readlib(iscont):
> iscont(h(x), x=-1..1);
```

$$\mathit{false}$$

```
> iscont(h(x), x=0..1);
```

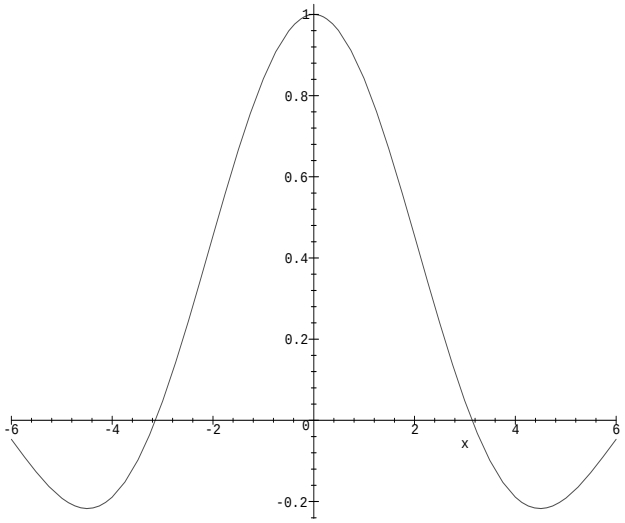
$$true$$

iscont muß zunächst geladen werden und erwartet als zweites Argument ein Intervall.
Wir betrachten nun die Funktion

```
> g := x -> sin(x)/x;
```

$$g := x \to \frac{\sin(x)}{x}$$

```
> plot(g(x), x=-6..6);
```




```
> g(0);
```

```
Error, (in g) division by zero
```

```
> iscont(g(x), x=-1..1);
```

$$true$$

Die Funktion g kann an der Stelle x=0 nicht ausgewertet werden (Ausdruck 0/0). Der Graph von g ist aber in der Umgebung von 0 offenbar stetig:

```
> Limit(g(x), x=0, left): % = value(%);
```

$$\lim_{x \to 0-} \frac{\sin(x)}{x} = 1$$

```
> Limit(g(x), x=0, right): % = value(%);
```

$$\lim_{x \to 0+} \frac{\sin(x)}{x} = 1$$

Der links- und rechtsseitige Grenzwert für x -> 0 sind gleich, d.h. die Funktion g ist an der Stelle x=0 stetig ergänzbar.

```
> g(0) := 1;
```

$$g(0) := 1$$

Durch diese Zuordnung wird ein Eintrag in die **remember table** von g vorgenommen. Wird nun die Funktion (Prozedur) g mit dem Argument 0 aufgerufen, dann wird g nicht an der Stelle 0 ausgewertet, sondern es wird der gespeicherte Funktionswert aus der remember table zurückgegeben. Den Inhalt der remember table erhält man mit

```
> op(4, op(g));
```

$$\begin{array}{l} \mathrm{table}([ \\ \quad 0 = 1 \\ \quad ]) \end{array}$$

```
> g(0);
```

$$1$$

Die remember table wird später bei der Einführung von Maple-Prozeduren noch genauer besprochen. Die Rechenregeln für Grenzwerte lassen sich mit der (trägen) Funktion Limit

(**student**,**Limit**) aus dem Maple-Paket student und der Funktion expand formal herleiten.

```
> restart:
> with(student):
```

Regel für multiplikative Konstanten:

```
> Limit(c*f(x),x=a): % = expand(%);
```

$$\lim_{x \to a} c\,\mathrm{f}(x) = c\,(\lim_{x \to a} \mathrm{f}(x))$$

Beispiel:

```
> subs(f(x)=x^2, c=2, a=1, %);
```

$$\lim_{x \to 1} 2\,x^2 = 2\,(\lim_{x \to 1} x^2)$$

```
> value(%);
```

$$2 = 2$$

Summenregel:

```
> Limit(f(x)+g(x), x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x) + \mathrm{g}(x) = (\lim_{x \to a} \mathrm{f}(x)) + (\lim_{x \to a} \mathrm{g}(x))$$

Beispiel:

```
> subs(f(x)=x^2, g(x)=x, a=2, %);
```

$$\lim_{x \to 2} x^2 + x = (\lim_{x \to 2} x^2) + (\lim_{x \to 2} x)$$

```
> value(%);
```

$$6 = 6$$

Produktregel:

```
> Limit(f(x)*g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x)\,\mathrm{g}(x) = (\lim_{x \to a} \mathrm{f}(x))\,(\lim_{x \to a} \mathrm{g}(x))$$

Beispiel:

```
> subs(f(x)=x-2, g(x)=x^2, a=1, %);
```

$$\lim_{x \to 1} (x-2)\,x^2 = (\lim_{x \to 1} x - 2)\,(\lim_{x \to 1} x^2)$$

```
> value(%);
```

$$-1 = -1$$

Quotientenregel (limit(g(x), x=a)<> 0):

```
> Limit(f(x)/g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \frac{\mathrm{f}(x)}{\mathrm{g}(x)} = \frac{\lim_{x \to a} \mathrm{f}(x)}{\lim_{x \to a} \mathrm{g}(x)}$$

Beispiel:

```
> subs(f(x)=x, g(x)=x^2+1, a=2, %);
```

$$\lim_{x \to 2} \frac{x}{x^2+1} = \frac{\lim_{x \to 2} x}{\lim_{x \to 2} x^2+1}$$

```
> value(%);
```

$$\frac{2}{5} = \frac{2}{5}$$

Regel für Potenzen:

```
> Limit(f(x)^g(x),x=a): % = expand(%);
```

$$\lim_{x \to a} \mathrm{f}(x)^{\mathrm{g}(x)} = (\lim_{x \to a} \mathrm{f}(x))^{(\lim_{x \to a}\ \mathrm{g}(x))}$$

Beispiel:

```
> subs(f(x)=x+2, g(x)=x, a=2, %);
```

$$\lim_{x \to 2} (x+2)^x = (\lim_{x \to 2} x + 2)^{(\lim_{x \to 2}\ x)}$$

```
> value(%);
```

$$16 = 16$$

Regel für Ungleichungen: f(x) $<$ g(x) => lim(f(x), x=a) $<$ lim(g(x), x=a), falls die Grenzwerte existieren. Beispiel:

```
> f:=x->x^2+3*x-1;
```

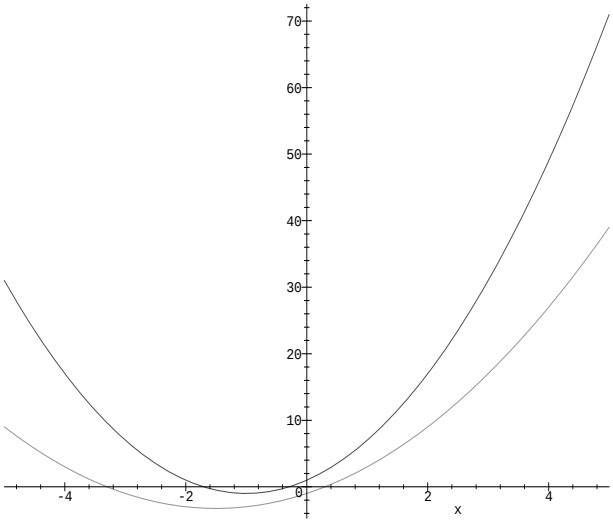
$$f := x \to x^2 + 3\,x - 1$$

```
> g:=x->2*x^2+4*x+1;
```

$$g := x \to 2\,x^2 + 4\,x + 1$$

```
> plot({f(x),g(x)},x=-5..5, title='f(x) < g(x) fuer alle x aus IR');
```

f(x) < g(x) fuer alle x aus IR




Für alle reellen a gilt

```
> f(a) < g(a);
```

$$a^2 + 3\,a - 1 < 2\,a^2 + 4\,a + 1$$

Also haben wir für die Grenzwerte

```
> Limit(f(x), x=a) < Limit(g(x), x=a);
```

$$\lim_{x \to a} x^2 + 3\,x - 1 < \lim_{x \to a} 2\,x^2 + 4\,x + 1$$

```
> value(%);
```

$$a^2 + 3\,a - 1 < 2\,a^2 + 4\,a + 1$$

```
> subs(a=4, %);
```

$$27 < 49$$

## 3.3 Differentiation

```
> restart:
```

Zunächst wollen wir die graphische Bedeutung der Ableitung einer Funktion f an der Stelle $x_0$ illustrieren. Dazu definieren wir die Funktion

```
> f := x -> exp(sin(x));
```

$$f := x \to e^{\sin(x)}$$

und bestimmen die Steigung der Tangente an die Kurve von f im Punkt [ $x_0$, f( $x_0$)] mit $x_0$=1.

```
> x[0] := 1;
```

$$x_0 := 1$$

Wir betrachten die Punkte

```
> p[0] := [x[0], f(x[0])];
```

$$p_0 := [1,\, e^{\sin(1)}]$$

und

```
> p[1] := [x[0] +h , f(x[0] + h)];
```

$$p_1 := [1 + h,\, e^{\sin(1+h)}]$$

auf dem Graphen von f. Die Steigung der Sekante durch $p_0$ und $p_1$ wird mit dem Kommando slope aus dem Maple-Paket student bestimmt (**student**,**slope**)

```
> with(student):
> m := slope(p[0], p[1]);
```

$$m := -\frac{e^{\sin(1)} - e^{\sin(1+h)}}{h}$$

Z.B. ist die Steigung für h=1

```
> subs(h=1, m);
```

$$-e^{\sin(1)} + e^{\sin(2)}$$

```
> evalf(%);
```

$$.162800903$$

Die Sekante wird durch folgende lineare Gleichung beschrieben:

```
> y - p[0][2] = m*(x - p[0][1]);
```

$$y - e^{\sin(1)} = -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h}$$

Mit dem **isolate** Kommando isolieren wir y auf der linken Seite der Gleichung und erhalten

```
> isolate(%, y);
```

$$y = -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h} + e^{\sin(1)}$$

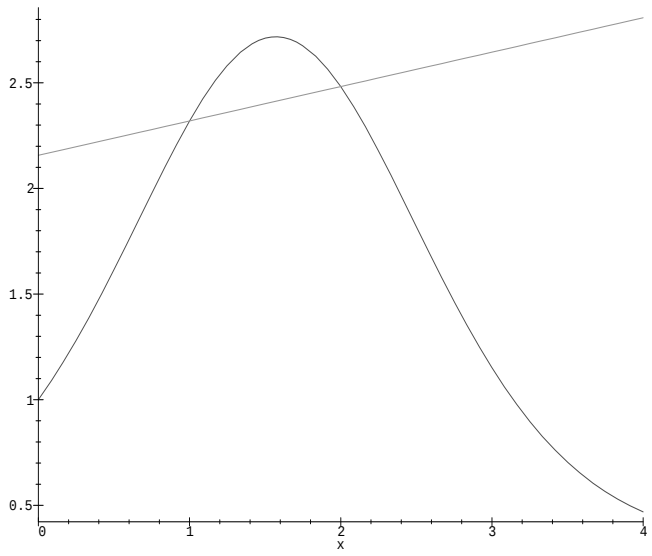
Die Sekante als Funktion von x und h läßt sich nun wie folgt definieren:

```
> s := unapply (rhs(%), x, h);
```

$$s := (x,\, h) \to -\frac{(e^{\sin(1)} - e^{\sin(1+h)})\,(x-1)}{h} + e^{\sin(1)}$$

Wir zeichnen jetzt die Funktionen f und s (mit h=1) in ein gemeinsames Schaubild.

```
> plot({f(x), s(x, 1)}, x=0..4);
```




Die Schnittpunkte von f und s sind hier gegeben durch

```
> 'p[0]' = evalf(p[0]);
```

$$p_0 = [1., 2.319776825]$$

und

```
> 'p[1]' = evalf(subs(h=1, p[1]));
```

$$p_1 = [2., 2.482577728]$$

Offenbar konvergiert die Steigung m für

```
> h_values := [seq(1/i, i=1..15)];
```

$$h\_values := [1, \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \frac{1}{6}, \frac{1}{7}, \frac{1}{8}, \frac{1}{9}, \frac{1}{10}, \frac{1}{11}, \frac{1}{12}, \frac{1}{13}, \frac{1}{14}, \frac{1}{15}]$$

```
> seq(evalf(m), h=h_values);
```

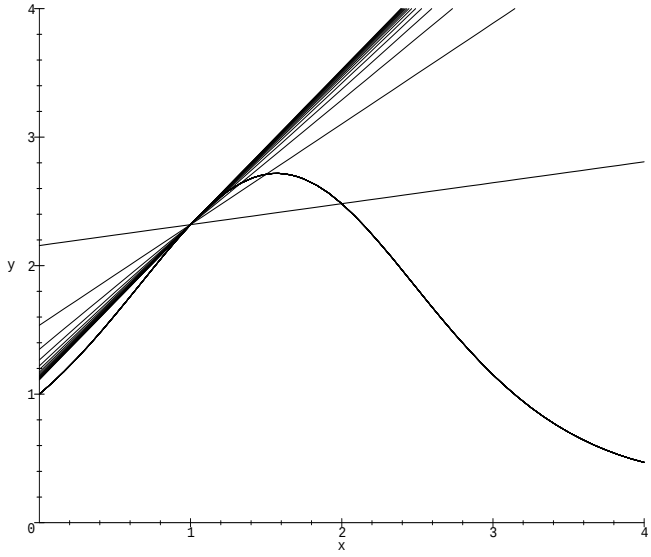
$$.162800903, .783408386, .969854001, 1.053234750, 1.09952853, 1.12872746, 1.14874064, 1.16327997, 1.17430578, 1.18294680, 1.18989715, 1.19560651, 1.20037862, 1.20442589, 1.20790129$$

Die Konvergenz der Sekantensteigung m wollen wir als Folge von 2D-Plots graphisch darstellen.

```
> S := seq(plot({f(x), s(x, h)}, x=0..4, y=0..4, color=black),
h=h_values):
```

Mit dem display Kommando aus dem Maple-Paket plots (**plots,display**) läßt sich diese Plotfolge in ein Schaubild zeichnen.

```
> with(plots):

> display([S]);
```

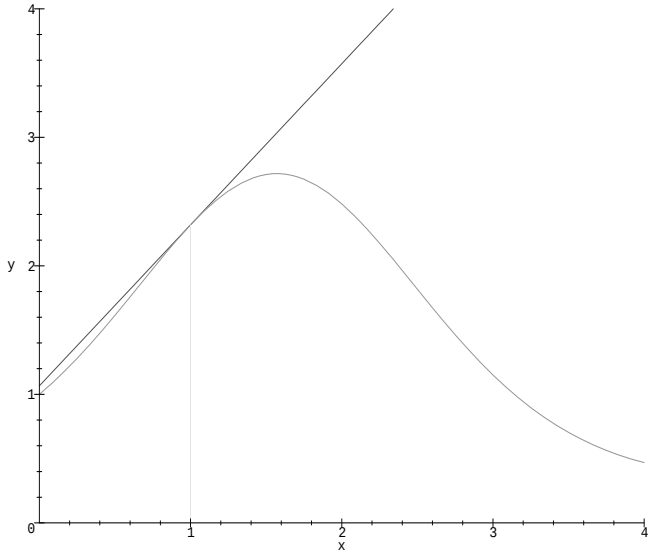



Die Option insequence=true ermöglicht die Animation dieser Plotfolge.

```
> display([S], insequence=true);
```

Mit dem Kommando **showtangent** aus dem Paket student läßt sich zu einer gegebenen Stelle x die Tangente an die Kurve f zeichnen.

```
> showtangent(f(x), x=1, x=0..4, y=0..4);
```




Im Grenzwert ist die Steigung

```
> Limit(m, h=0);
```

$$\lim_{h \to 0} -\frac{e^{\sin(1)} - e^{\sin(1+h)}}{h}$$

```
> value(%);
```

$$e^{\sin(1)} \cos(1)$$

Dies ist natürlich die Ableitung von f an der Stelle $x_0$=1.

```
> D(f)(x[0]);
```

$$e^{\sin(1)}\cos(1)$$

Allgemein ist die Ableitung von f an der Stelle $x_0$ definiert durch den folgenden Grenzwert:

```
> f := 'f': x[0] := 'x[0]':
> Limit((f(x[0]+h)-f(x[0]))/h, h=0);
```

$$\lim_{h\to 0}\frac{\mathrm{f}(x_0+h)-\mathrm{f}(x_0)}{h}$$

```
> value(%);
```

$$\mathrm{D}(f)(x_0)$$

Die Ableitung als Funktion von x erhält man mit dem **D**-Operator.

```
> f := x -> exp(sin(x));
```

$$f := x \to e^{\sin(x)}$$

```
> `f'` := D(f);
```

$$f' := x \to \cos(x)\, e^{\sin(x)}$$

Die Differentiation eines Maple-Ausdrucks (einer Formel) erfolgt mit dem Kommando **diff**

```
> Diff (f(x), x) = diff (f(x), x);
```

$$\frac{\partial}{\partial x}\, e^{\sin(x)} = \cos(x)\, e^{\sin(x)}$$

**Diff** ist hier wieder die träge Form des diff-Kommandos. Die Ableitungsfunktion erhält man hieraus mittels

```
> f1 := unapply(rhs(%), x);
```

$$f1 := x \to \cos(x)\, e^{\sin(x)}$$

Die zweite Ableitung von f erhalten wir durch Differenzieren von f1

```
> diff(f1(x), x);
```

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

oder direkt mit

```
> diff(f(x), x, x);
```

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

oder durch

```
> diff(f(x), x$2);
```

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

Hierbei ist

```
> x$2;
```

$$x,\ x$$

Mit Hilfe des D-Operators läßt sich die zweite Ableitung von f durch

```
> D(D(f))(x);
```

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

oder unter Verwendung des Operators @für die Komposition (Hintereinanderschaltung) von Funktionen vermöge

```
> (D@@2)(f)(x);
```

berechnen.

$$-\sin(x)\, e^{\sin(x)} + \cos(x)^2\, e^{\sin(x)}$$

## 3.4 Differentiationsregeln

```
> restart:
```

Wir wollen nun die bekannten Differentiationsregeln mit Hilfe der Maple-Funktion **limit** ableiten. Zunächst betrachten wir die einfache Potenzfunktion

```
> f := x -> x^n;
```

$$f := x \to x^n$$

und berechnen den Grenzwert des Differenzenquotienten für $x \to x_0$ bzw. $h \to 0$.

```
> Limit((f(x[0]+h)-f(x[0]))/h, h=0): % = simplify(value(%));
```

$$\lim_{h \to 0} \frac{(x_0 + h)^n - {x_0}^n}{h} = {x_0}^{(-1+n)}\, n$$

Für die Potenzfunktion kennt die limit Funktion also den Grenzwert des Differenzenquotienten. Allgemein gilt für x aus IR

```
> Diff(f(x), x) = simplify(diff(f(x), x));
```

$$\frac{\partial}{\partial x} x^n = x^{(-1+n)}\, n$$

Die Ableitungsfunktion ergibt sich daraus wie folgt:

```
> `f'` := unapply(rhs(%), x);
```

$$f' := x \to x^{(-1+n)}\, n$$

Nun betrachten wir unbestimmte Funktionen auf IR. Seien die Funktionen f und g überall differenzierbar. Dann gilt für die Funktion f+g an der Stelle $x_0$

```
> f := 'f':
> Limit((f(x[0]+h)+g(x[0]+h)-f(x[0])-g(x[0]))/h, h=0): % = value(%);
```

$$\lim_{h \to 0} \frac{\mathrm{f}(x_0 + h) + \mathrm{g}(x_0 + h) - \mathrm{f}(x_0) - \mathrm{g}(x_0)}{h} = \mathrm{D}(f)(x_0) + \mathrm{D}(g)(x_0)$$

Limit kennt also auch den Grenzwert des Differenzenquotienten für die Summenfunktion f+g. Allgemein haben wir die folgende Summenregel

```
> 'D(f+g)(x)' = D(f+g)(x);
```

$$\mathrm{D}(f + g)(x) = \mathrm{D}(f)(x) + \mathrm{D}(g)(x)$$

oder in der Leibniz-Notation

```
> Diff((f+g)(x), x): % = value(%);
```

$$\frac{\partial}{\partial x}(\mathrm{f}(x) + \mathrm{g}(x)) = (\frac{\partial}{\partial x}\mathrm{f}(x)) + (\frac{\partial}{\partial x}\mathrm{g}(x))$$

Für die Ableitung der Funktion f*g an der Stelle $x_0$ gilt

```
> Limit((f(x[0]+h)*g(x[0]+h)-f(x[0])*g(x[0]))/h, h=0): % = value(%);
```

$$\lim_{h \to 0} \frac{\mathrm{f}(x_0 + h)\,\mathrm{g}(x_0 + h) - \mathrm{f}(x_0)\,\mathrm{g}(x_0)}{h} = \mathrm{f}(x_0)\,\mathrm{D}(g)(x_0) + \mathrm{D}(f)(x_0)\,\mathrm{g}(x_0)$$

Allgemein gilt die Produktregel

```
> 'D(f*g)(x)' = D(f*g)(x);
```

$$\mathrm{D}(f\,g)(x) = \mathrm{D}(f)(x)\,\mathrm{g}(x) + \mathrm{f}(x)\,\mathrm{D}(g)(x)$$

Die Ableitung der Funktion f/g an der Stelle $x_0$ ( $\mathrm{g}(x_0) \neq 0$) läßt sich mit der Formel

```
> Limit((f(x[0]+h)/g(x[0]+h)-f(x[0])/g(x[0]))/h, h=0): % = value(%);
```

$$\lim_{h \to 0} \frac{\dfrac{\mathrm{f}(x_0 + h)}{\mathrm{g}(x_0 + h)} - \dfrac{\mathrm{f}(x_0)}{\mathrm{g}(x_0)}}{h} = \frac{\mathrm{D}(f)(x_0)\,\mathrm{g}(x_0) - \mathrm{f}(x_0)\,\mathrm{D}(g)(x_0)}{\mathrm{g}(x_0)^2}$$

berechnen. Allgemein gilt die Quotientenregel

```
> 'D(f/g)(x)' = normal(D(f/g)(x));
```

$$\mathrm{D}(\frac{f}{g})(x) = -\frac{-\mathrm{D}(f)(x)\,\mathrm{g}(x) + \mathrm{f}(x)\,\mathrm{D}(g)(x)}{\mathrm{g}(x)^2}$$

Für die Ableitung der geschachtelten Funktion f@g an der Stelle $x_0$ gilt

```
> Limit((f(g(x+h))-f(g(x)))/h, h=0): % = value(%);
```

$$\lim_{h \to 0} \frac{\mathrm{f}(\mathrm{g}(x + h)) - \mathrm{f}(\mathrm{g}(x))}{h} = \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

und somit allgemein die Kettenregel

```
> 'D(f@g)(x)' = D(f@g)(x);
```

$$\mathrm{D}(f@g)(x) = \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

## 3.5 Kurvendiskussion

```
> restart:
```

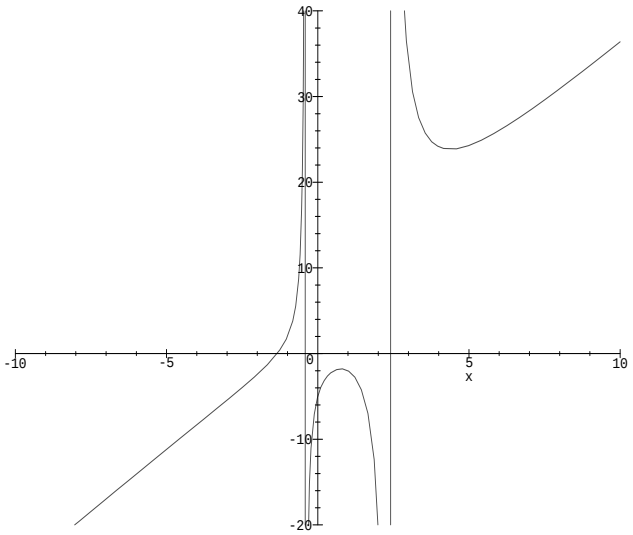
Mit den Hilfsmitteln der Differentialrechnung lassen sich die Eigenschaften und Kennzeichen nichtlinearer Kurven bestimmen. Als Beispiel wollen wir den Graph der rationalen Funktion

```
> f := x -> (3*x^3-x^2-3*x+5) / (x^2-2*x-1);
```

$$f := x \to \frac{3\,x^3 - x^2 - 3\,x + 5}{x^2 - 2\,x - 1}$$

diskutieren. Zunächst plotten wir die Funktion f.

```
> plot(f(x), x=-10..10, -20 .. 40);
```




Der Funktionsgraph besteht offenbar aus drei Teilstücken, die durch Asymptoten begrenzt werden. Es gibt zwei Polstellen (vertikale Asymptoten). Durch Berechnung der Nullstellen des Nennerpolynoms (**denom**) bestimmen wir diese Polstellen.

```
> Pole := solve(denom(f(x)) = 0, x);
```

$$Pole := 1 + \sqrt{2},\ 1 - \sqrt{2}$$

```
> evalf(Pole);
```

$$2.414213562,\ -.414213562$$

Neben den beiden vertikalen Asymptoten (Polstellen) gibt es noch eine weitere nicht-vertikale Asymptote, die sich durch Berechnung des Quotienten (**quo**) aus Zähler- und Nennerpolynom ermitteln läßt.

```
> q := quo(numer(f(x)), denom(f(x)), x, 'r');
```

$$q := 3\,x + 5$$

Der Rest der Polynomdivision steht in der Variablen

```
> r;
```

$$10 + 10\,x$$

zur Verfügung, spielt aber für die weiteren Betrachtungen keine Rolle.

```
> q := unapply(q, x);
```

$$q := x \to 3\,x + 5$$

Die rationale Funktion f verhält sich also für große x wie die lineare Funktion q. Diese Tatsache wird auch durch die asymptotische Entwicklung von f(x) bestätigt. Die Anwendung des Maple-Kommandos **asympt** liefert nämlich

```
> asympt(f(x), x);
```

$$3\,x + 5 + 10\,\frac{1}{x} + 30\,\frac{1}{x^2} + 70\,\frac{1}{x^3} + 170\,\frac{1}{x^4} + 410\,\frac{1}{x^5} + \mathrm{O}(\frac{1}{x^6})$$

Mit dem limit Kommando läßt sich das asymptotische Verhalten von f(x) für große x verifizieren.

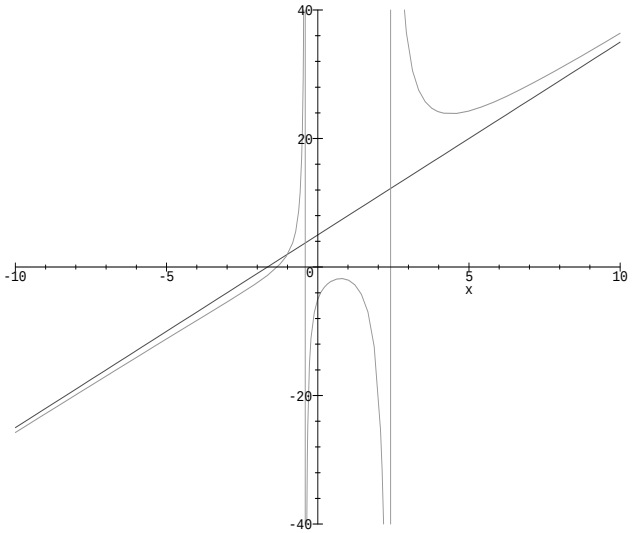
```
> Limit('f(x)/q(x)', x=infinity) = limit(f(x)/q(x), x=infinity);
```

$$\lim_{x\to\infty} \frac{\mathrm{f}(x)}{\mathrm{q}(x)} = 1$$

```
> Limit('f(x)/q(x)', x=-infinity) = limit(f(x)/q(x), x=-infinity);
```

$$\lim_{x\to(-\infty)} \frac{\mathrm{f}(x)}{\mathrm{q}(x)} = 1$$

```
> plot({f(x), q(x)}, x=-10..10, -40 .. 40);
```




Nun bestimmen wir die Schnittstelle des Graphen mit der x-Achse. Dazu müssen die Nullstellen der Funktion f berechnet werden.

```
> f_null := solve(f(x) = 0, x);
```

$$f\_null := -\frac{1}{9}\,\%1^{(1/3)} - \frac{28}{9}\,\frac{1}{\%1^{(1/3)}} + \frac{1}{9},$$
$$\frac{1}{18}\,\%1^{(1/3)} + \frac{14}{9}\,\frac{1}{\%1^{(1/3)}} + \frac{1}{9} + \frac{1}{2}\,I\,\sqrt{3}\,(-\frac{1}{9}\,\%1^{(1/3)} + \frac{28}{9}\,\frac{1}{\%1^{(1/3)}}),$$
$$\frac{1}{18}\,\%1^{(1/3)} + \frac{14}{9}\,\frac{1}{\%1^{(1/3)}} + \frac{1}{9} - \frac{1}{2}\,I\,\sqrt{3}\,(-\frac{1}{9}\,\%1^{(1/3)} + \frac{28}{9}\,\frac{1}{\%1^{(1/3)}})$$
$$\%1 := 566 + 18\,\sqrt{921}$$

Zwei dieser Lösungen sind komplex:

```
> evalf(f_null);
```

$$-1.340384213,\ .8368587730 - .7369477385\,I,\ .8368587730 + .7369477385\,I$$

Der Schnittpunkt des Graphen mit der x-Achse wird durch die reelle Nullstelle bestimmt.

```
> pts := x -> [x, f(x)]:
> simplify(pts(f_null[1]));
```

$$[-\frac{1}{9}\,\frac{(566 + 18\,\sqrt{921})^{(2/3)} + 28 - (566 + 18\,\sqrt{921})^{(1/3)}}{(566 + 18\,\sqrt{921})^{(1/3)}},\ 0]$$

```
> evalf(%);
```

$$[-1.340384212,\ 0]$$

Der Schnittpunkt des Funktionsgraphen mit der y-Achse ist durch

```
> pts(0);
```

$$[0,\ -5]$$

gegeben. Die Asymptote q schneidet die Funktion f im Punkt

```
> pts(solve(f(x) = q(x), x));
```

$$[-1,\ 2]$$

Nun bestimmen wir die relativen Extrema und Wendepunkte des Funktionsgraphen. Hierzu benötigen wir die ersten drei Ableitungen der Funktion f.

```
> diff(f(x), x);
```

$$\frac{9\,x^2-2\,x-3}{x^2-2\,x-1}-\frac{(3\,x^3-x^2-3\,x+5)\,(2\,x-2)}{(x^2-2\,x-1)^2}$$

```
> f1 := unapply(normal(%), x);
```

$$f1 := x \to \frac{3\,x^4-12\,x^3-4\,x^2-8\,x+13}{(x^2-2\,x-1)^2}$$

```
> diff(f(x), x$2);
```

$$\frac{18\,x-2}{x^2-2\,x-1}-2\,\frac{(9\,x^2-2\,x-3)\,(2\,x-2)}{(x^2-2\,x-1)^2}+2\,\frac{(3\,x^3-x^2-3\,x+5)\,(2\,x-2)^2}{(x^2-2\,x-1)^3}-2\,\frac{3\,x^3-x^2-3\,x+5}{(x^2-2\,x-1)^2}$$

```
> f2 := unapply(normal(%), x);
```

$$f2 := x \to 20\,\frac{x^3+3\,x^2-3\,x+3}{(x^2-2\,x-1)^3}$$

```
> diff(f(x), x$3);
```

$$18\,\frac{1}{x^2-2\,x-1}-3\,\frac{(18\,x-2)\,(2\,x-2)}{(x^2-2\,x-1)^2}+6\,\frac{(9\,x^2-2\,x-3)\,(2\,x-2)^2}{(x^2-2\,x-1)^3}-6\,\frac{9\,x^2-2\,x-3}{(x^2-2\,x-1)^2}-6\,\frac{(3\,x^3-x^2-3\,x+5)\,(2\,x-2)^3}{(x^2-2\,x-1)^4}+12\,\frac{(3\,x^3-x^2-3\,x+5)\,(2\,x-2)}{(x^2-2\,x-1)^3}$$

```
> f3 := unapply(normal(%), x);
```

$$f3 := x \to -60\,\frac{12\,x+4\,x^3-6\,x^2+x^4-7}{(x^2-2\,x-1)^4}$$

Notwendig für das Vorhandensein eines Extremums ist das Verschwinden der ersten Ableitung. Der Zähler von f1 ist ein Polynom vierten Grades. Da die expliziten Lösungen einer polynomialen Gleichung vierten Grades im allgemeinen sehr komplex sind, werden sie in Maple nur dann ausgegeben, wenn man die globale Variable _EnvExplict vorher auf true setzt (**solve**).

```
> _EnvExplicit := true;
```

$$\_EnvExplicit := true$$

```
> solve(f1(x) = 0, x);
```

$$1+\frac{1}{2}\sqrt{10}+\frac{1}{6}\sqrt{42+24\sqrt{10}},\ 1+\frac{1}{2}\sqrt{10}-\frac{1}{6}\sqrt{42+24\sqrt{10}},\ 1-\frac{1}{2}\sqrt{10}+\frac{1}{6}\sqrt{42-24\sqrt{10}},\ 1-\frac{1}{2}\sqrt{10}-\frac{1}{6}\sqrt{42-24\sqrt{10}}$$

```
> evalf(%);
```

$$4.390793984,\ .771483676,\ -.581138830+.9703187314\,I,\ -.581138830-.9703187314\,I$$

Die erste Ableitung hat also zwei reelle Nullstellen, die wir mit dem **select** Kommando aus der Lösungsmenge selektieren (select erwartet als zweites Argument eine Liste oder eine Menge).

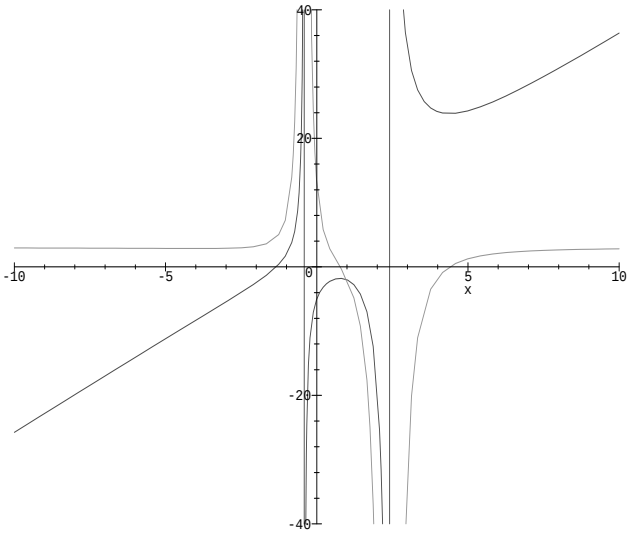
```
> f1_null := select(type, [%], realcons);
```

$$f1\_null := [4.390793984, .771483676]$$

```
> f1_null := sort(%);
```

$$f1\_null := [.771483676, 4.390793984]$$

```
> plot({f1(x), f(x)}, x=-10..10, -40 .. 40);
```




Die zweite Ableitung f2 hat an diesen Stellen die Werte

```
> f2(f1_null[1]);
```

$$-7.930860162$$

```
> f2(f1_null[2]);
```

$$3.088974054$$

Also haben wir im Punkt

```
> pts(f1_null[1]);
```

$$[.771483676, -1.780433469]$$

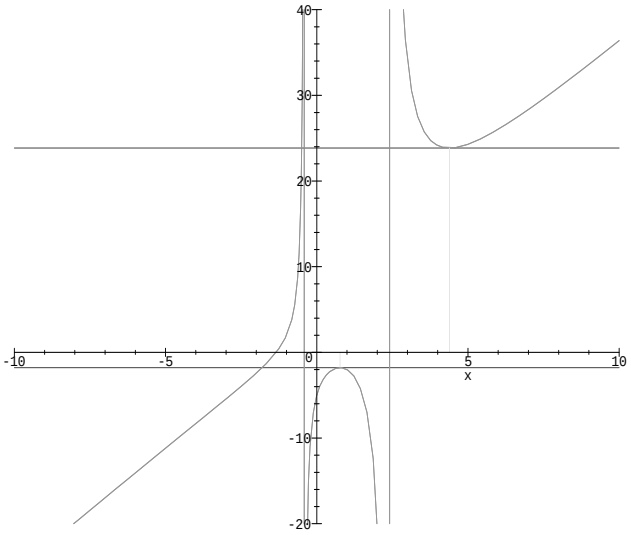
ein relatives Maximum und im Punkt

```
> pts(f1_null[2]);
```

$$[4.390793984, 23.84840530]$$

ein relatives Minimum.

```
> with(student): with(plots):
> p1 := showtangent(f(x), x=f1_null[1], x=-10..10, -20 .. 40):
> p2 := showtangent(f(x), x=f1_null[2], x=-10..10, -20 .. 40):
> display({p1, p2});
```




Notwendig für das Vorhandensein eines Wendepunktes ist das Verschwinden der zweiten Ableitung. Die Nullstellen der zweiten Ableitung sind

```
> solve(f2(x) = 0, x);
```

$$-\%2 - 2\,\%1 - 1,\ \frac{1}{2}\,\%2 + \%1 - 1 + \frac{1}{2}\,I\,\sqrt{3}\,(-\%2 + 2\,\%1),$$
$$\frac{1}{2}\,\%2 + \%1 - 1 - \frac{1}{2}\,I\,\sqrt{3}\,(-\%2 + 2\,\%1)$$
$$\%1 := \frac{1}{(4 + 2\,\sqrt{2})^{(1/3)}}$$
$$\%2 := (4 + 2\,\sqrt{2})^{(1/3)}$$

```
> evalf(%);
```

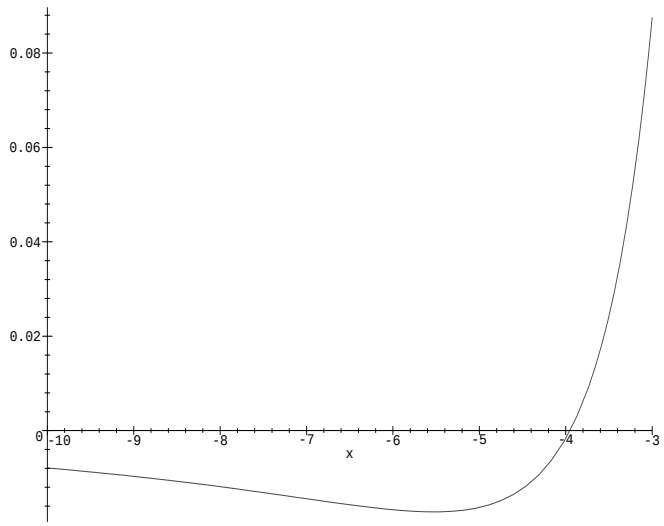
$$-3.951373036,\ .475686518 - .7300356815\,I,\ .475686518 + .7300356815\,I$$

Es gibt also nur eine reelle Nullstelle, nämlich

```
> f2_null := op(select(type, [%], realcons));
```

$$f2\_null := -3.951373036$$

```
> plot(f2(x), x=-10..-3);
```




Die dritte Ableitung hat an dieser Stelle den Wert

```
> f3(f2_null);
```

$$.03527228299$$

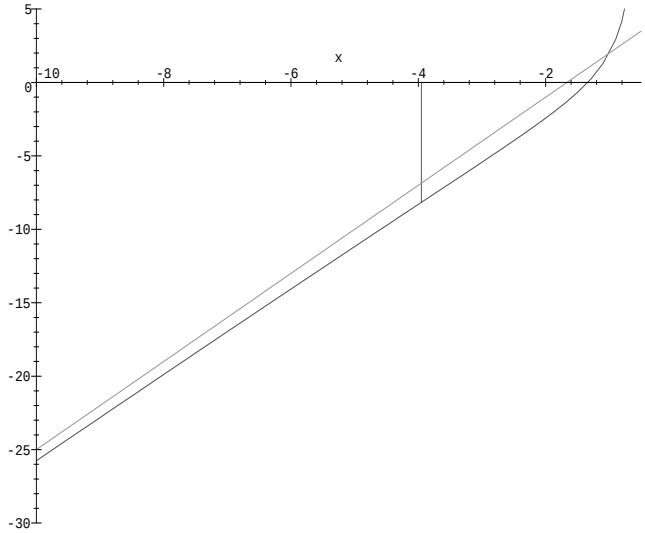
Also haben wir im Punkt

```
> pts(f2_null);
```

$$[-3.951373036,\ -8.164902817]$$

einen Wendepunkt.

```
> p3 := plot([[f2_null, 0], pts(f2_null)]):
> p4 := plot({f(x), q(x)}, x=-10..-0.5, -30..5):
> display({p3, p4});
```



## 3.6 Extremwertprobleme

```
> restart:
```

Die Lösung von Extremwertproblemen ist ein weiteres Anwendungsfeld der Differentialrechnung. Es folgt ein einfaches Beispiel: Es soll eine zylindrische Dose aus Blech mit vorgegebenem Inhalt (z.B. 1 Liter) hergestellt werden. Um den Blechverbrauch zu minimieren, wird die Dose mit der kleinsten Oberfläche gesucht.

Oberfläche und Volumen eines Zylinders sind durch die Gleichungen

```
> eqn1 := A = 2*Pi*r^2 + 2*Pi*r*h: %;
```

$$A = 2\,\pi\,r^2 + 2\,\pi\,r\,h$$

und

```
> eqn2 := V = Pi*r^2*h: %;
```

$$V = \pi\,r^2\,h$$

gegeben. Hierbei ist r der Radius und h die Höhe des Zylinders. Wir eliminieren nun die Höhe h in der Gleichung für die Oberfläche mit Hilfe der zweiten Beziehung, der Volumengleichung:

```
> solve(eqn2, {h});
```

$$\{h = \frac{V}{\pi\,r^2}\}$$

```
> assign(%);
```

Mit dem **assign** Kommando wird die Lösung der Variablen h zugewiesen.

```
> h;
```

$$\frac{V}{\pi\,r^2}$$

```
> eqn1;
```

$$A = 2\,\pi\,r^2 + 2\,\frac{V}{r}$$

Die Zylinderoberfläche als Funktion von r ist damit gegeben durch

```
> A := unapply(rhs(%), r);
```

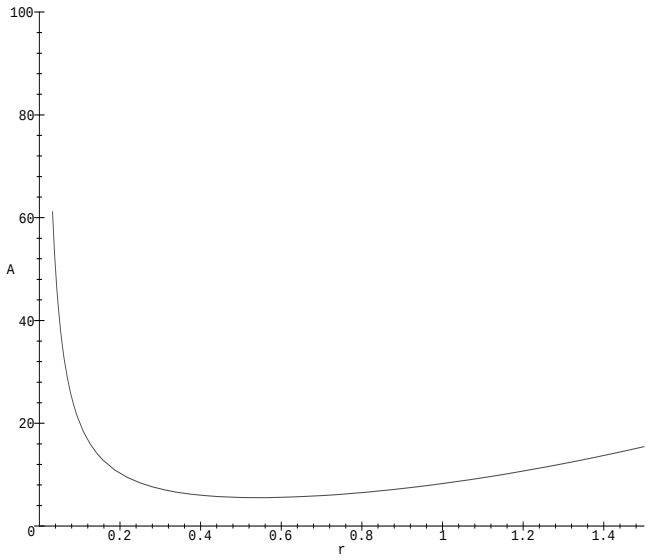
$$A := r \to 2\,\pi\,r^2 + 2\,\frac{V}{r}$$

Bei gegebenem Volumen muß nun die Oberfläche minimiert werden.

```
> V := 1;
```

$$V := 1$$

```
> plot(A(r), r=0..1.5, A=0..100);
```



In der Umgebung von 0.5 befindet sich offenbar ein Minimum. Wir bestimmen nun die Nullstellen der ersten Ableitung.

```
> A1 := D(A);
```

$$A1 := r \to 4\,\pi\,r - 2\,\frac{V}{r^2}$$

```
> A1_null := simplify([solve(A1(r) = 0, r)]);
```

$$A1\_null := \left[\frac{1}{2}\,\frac{2^{(2/3)}}{\pi^{(1/3)}},\ \frac{1}{4}\,\frac{2^{(2/3)}\,(-1+I\,\sqrt{3})}{\pi^{(1/3)}},\ -\frac{1}{4}\,\frac{2^{(2/3)}\,(1+I\,\sqrt{3})}{\pi^{(1/3)}}\right]$$

```
> evalf(%);
```

$$[.5419260700,\ -.2709630350 + .4693217438\,I,\ -.2709630350 - .4693217438\,I]$$

Es existiert eine reelle Nullstelle. Die zweite Ableitung ist an dieser Stelle

```
> D(A1)(A1_null[1]);
```

$$12\,\pi$$

positiv, d.h. der Funktionsgraph hat im Punkt

```
> simplify([A1_null[1], A(A1_null[1])]);
```

$$\left[\frac{1}{2}\,\frac{2^{(2/3)}}{\pi^{(1/3)}},\ 3\,\pi^{(1/3)}\,2^{(1/3)}\right]$$

```
> evalf(%);
```

$$[.5419260700,\ 5.535810447]$$

ein Minimum. Die zylindrische Dose hat also bei

```
> r := A1_null[1]: 'r' = evalf(%);
```

$$r = .5419260700$$

und

```
> 'h' = evalf(h);
```

$$h = 1.083852140$$

die minimale Oberfläche

```
> A = evalf(A(r));
```

$$A = 5.535810447$$

# Kapitel 4

# Integralrechnung

## 4.1 Riemann-Integral

```
> restart:
```

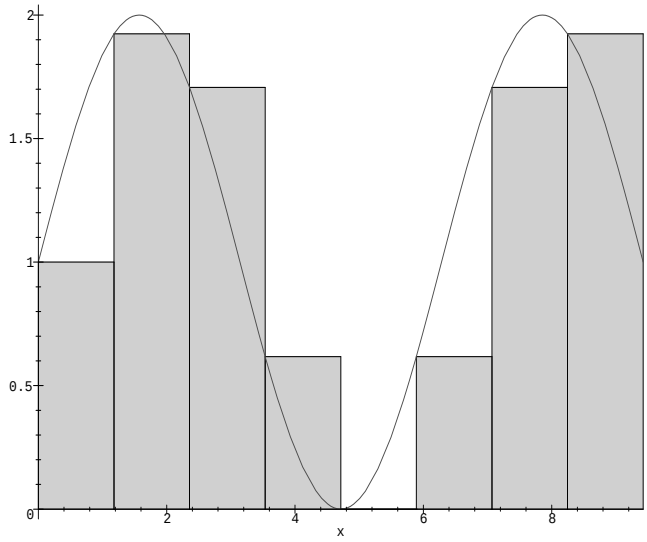
Das Riemann-Integral einer Funktion f mißt die Fläche zwischen der x-Achse und dem Funktionsgraphen. Diese geometrische Bedeutung des Integralbegriffs wollen wir nun mathematisch genauer interpretieren. Dazu definieren wir die Funktion

```
> f := x -> 1 + sin(x);
```

$$f := x \to 1 + \sin(x)$$

und zeichnen mit dem **leftbox** Kommando aus dem Paket student 8 "linksseitige" Rechtecke zwischen dem Graphen von f und der x-Achse.

```
> with(student):
> leftbox(f(x), x=0..3*Pi, 8);
```




Mit dem Kommando **leftsum** wird der Flächeninhalt der Rechtecke aufaddiert.

```
> leftsum(f(x), x=0..3*Pi, 8);
```

$$\frac{3}{8}\,\pi\,\left(\sum_{i=0}^{7}(1+\sin(\frac{3}{8}\,i\,\pi))\right)$$

Für die Fläche zwischen Kurve und x-Achse gilt also näherungsweise

```
> evalf(%);
```

$$11.18792509$$

Je mehr Rechtecke für die Approximation verwendet werden, desto besser wird die Näherung für die Fläche.

```
> boxes := [seq(n^2, n=1..12)];
```

$$boxes := [1, 4, 9, 16, 25, 36, 49, 64, 81, 100, 121, 144]$$

```
> seq(evalf(leftsum(f(x), x=0..3*Pi, n)), n=boxes);
```

$$9.424777962, 10.40074568, 11.23857733, 11.36661104, 11.40103461, 11.41334174, 11.41860822, 11.42116230, 11.42252103, 11.42329730, 11.42376670, 11.42406396$$

Die Konvergenz der Rechteckapproximationen wollen wir als Folge von 2D-Plots graphisch darstellen.

```
> S := seq(leftbox(f(x), x=0..3*Pi, n), n=boxes):
> with(plots):
> display([S], insequence=true);
```

Den Flächeninhalt erhalten wir als Grenzwert der Rechteckapproximationen. Die Approximation mit n Rechtecken liefert die Fläche

```
> A[n] :=leftsum(f(x), x=0..3*Pi, n);
```

$$A_n := 3\,\frac{\pi\left(\sum_{i=0}^{n-1}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n}$$

```
> combine(simplify(value(%)), trig);
```

$$\frac{3\,\pi\cos(2\,\frac{\pi}{n})+3\,\pi\,n\sin(2\,\frac{\pi}{n})+3\,\pi\cos(\frac{\pi}{n})+3\,\pi\,n\sin(\frac{\pi}{n})}{n\sin(\frac{\pi}{n})+n\sin(2\,\frac{\pi}{n})}$$

Für *n* -> $\infty$ ergibt sich

```
> Limit(A[n], n=infinity) = limit(%, n=infinity);
```

$$\lim_{n\to\infty} 3\,\frac{\pi\left(\sum_{i=0}^{n-1}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n} = 2+3\,\pi$$

Nun berechnen wir den Flächeninhalt mittels "rechtsseitiger" Rechteckapproximationen.

```
> B[n] := rightsum(f(x), x=0..3*Pi, n);
```

$$B_n := 3\,\frac{\pi\left(\sum_{i=1}^{n}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n}$$

```
> combine(simplify(value(%)), trig);
```

$$(3\,\pi\cos(\frac{\pi}{n})+3\,\pi\cos(2\,\frac{\pi}{n})+6\,\pi\,n\sin(\frac{\pi}{n})-3\,\pi\cos(\frac{\pi\,(3\,n+1)}{n})+6\,\pi\,n\sin(2\,\frac{\pi}{n}) - 3\,\pi\cos(\frac{\pi\,(3\,n+2)}{n}))\Big/(2\,n\sin(2\,\frac{\pi}{n})+2\,n\sin(\frac{\pi}{n}))$$

```
> Limit(B[n], n=infinity) = limit(%, n=infinity);
```

$$\lim_{n\to\infty} 3\,\frac{\pi\left(\sum_{i=1}^{n}(1+\sin(3\,\frac{i\,\pi}{n}))\right)}{n} = 2+3\,\pi$$

In entsprechender Weise läßt sich der Flächeninhalt mit middelsum durch "Mittelpunktsrechtecke" approximieren. Der eindeutige Grenzwert (Flächeninhalt) heißt bestimmtes Integral von f über dem Intervall [0, 3*Pi]. Bestimmte Integrale werden in Maple mit dem Kommando **int** berechnet:

```
> Int(f(x), x=0..3*Pi) = int(f(x), x=0..3*Pi) ;
```

$$\int_0^{3\,\pi} 1+\sin(x)\,dx = 2+3\,\pi$$

**Int** ist hier die träge Form des int Kommandos. Wir lassen nun die obere Grenze des Integrationsintervalls unbestimmt und erhalten

```
> Int(f(t), t=0..x) = int(f(t), t=0..x);
```

$$\int_0^x 1+\sin(t)\,dt = x-\cos(x)+1$$

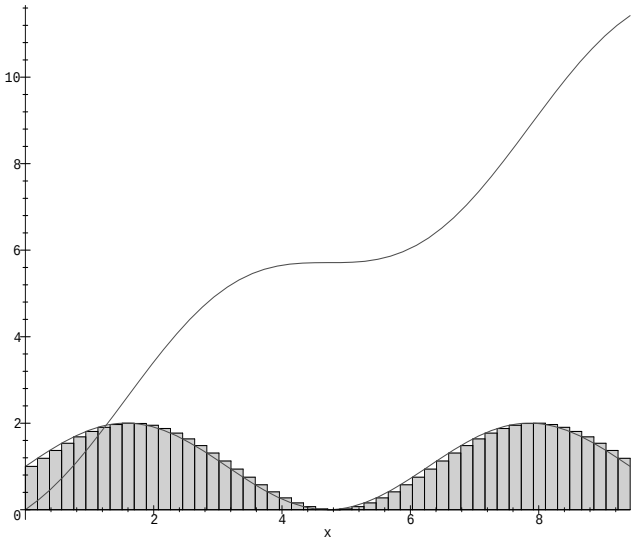
Die Funktion

```
> F := unapply(rhs(%), x);
```

$$F := x \to x-\cos(x)+1$$

heißt Stammfunktion von f. Wir plotten nun die Funktion F zusammen mit einer Rechteckappproximation in ein Schaubild.

```
> p1 := leftbox(f(x), x=0..3*Pi, 50):
> p2 := plot(F(x), x=0..3*Pi):
> display({p1, p2});
```




Diese Graphik zeigt, daß die Ableitung der Stammfunktion F (Änderungsrate der Flächenzunahme) durch die Funktion f beschrieben wird.

```
> D(F);
```

$$x \to 1+\sin(x)$$

Stammfunktionen sind nicht eindeutig, da additive Konstanten die Ableitung 0 haben.

```
> const := x -> c;
```

$$const := x \to c$$

```
> D(F+const);
```

$$x \to 1+\sin(x)$$

Die unbestimmte Integration mit der Maple-Prozedur int berücksichtigt keine Integrationskonstanten:

```
> Int(f(x), x) = int(f(x), x);
```

$$\int 1+\sin(x)\,dx = x-\cos(x)$$

Für die Berechnung eines bestimmten Integrals mittels Stammfunktion spielt die Integrationskonstante keine Rolle.

```
> F := unapply(rhs(%), x);
```

$$F := x \to x-\cos(x)$$

```
> Int(f(x), x=0..3*Pi) = F(3*Pi) - F(0);
```

$$\int_0^{3\,\pi} 1+\sin(x)\,dx = 2+3\,\pi$$

Allgemein gilt für stetige Funktionen f (Fundamentalsatz)

```
> f := 'f':
> F := x -> int(f(t), t=a..x);
```

$$F := x \to \int_a^x \mathrm{f}(t)\,dt$$

die Beziehung

```
> Diff(F(x), x) = diff(F(x), x);
```

$$\frac{\partial}{\partial x}\int_a^x \mathrm{f}(t)\,dt = \mathrm{f}(x)$$

oder in funktionaler Schreibweise

```
> 'D(F)' = D(F);
```

$$\mathrm{D}(F) = f$$

Die Ableitung der Stammfunktion von f ist also wieder f, d.h. mit der Definition des Differentialquotienten gilt

```
> Limit((F(x+h) - F(x))/h, h=0) = limit((F(x+h) - F(x))/h, h=0);
```

$$\lim_{h\to 0}\frac{\int_a^{x+h}\mathrm{f}(t)\,dt - \int_a^x \mathrm{f}(t)\,dt}{h} = \mathrm{f}(x)$$

Für die Berechnung bestimmter Integrale haben wir

```
> Int(f(t), t=c..d) = 'F'(d) - 'F'(c);
```

$$\int_c^d \mathrm{f}(t)\,dt = \mathrm{F}(d) - \mathrm{F}(c)$$

## 4.2 Elementare Integrationstechniken

```
> restart:
```

In diesem Abschnitt behandeln wir einige elementare Integrationstechniken. Die Technik der Variablensubstitution läßt sich auf die Ableitungsregel für zusammengesetzte Funktionen zurückführen (Kettenregel). Es gilt allgemein

```
> with(student):
> h := x -> f(g(x));
```

$$h := x \to \mathrm{f}(\mathrm{g}(x))$$

```
> h1 := D(h);
```

$$h1 := x \to \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)$$

```
> i1 := Int(h1(x), x);
```

$$i1 := \int \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)\,dx$$

```
> i2 := Int(h1(x), x=a..b);
```

$$i2 := \int_a^b \mathrm{D}(f)(\mathrm{g}(x))\,\mathrm{D}(g)(x)\,dx$$

Setzen wir nun g(x)=u, dann ist D(g)(x)dx = du und wir erhalten die transformierten Beziehungen mit **changevar**

```
> changevar(g(x)=u, i1, u): % = value(%);
```

$$\int \mathrm{D}(f)(u)\,du = \mathrm{f}(u)$$

bzw.

```
> changevar(g(x)=u, i2, u): % = value(%);
```

$$\int_{\mathrm{g}(a)}^{\mathrm{g}(b)} \mathrm{D}(f)(u)\,du = \mathrm{f}(\mathrm{g}(b)) - \mathrm{f}(\mathrm{g}(a))$$

Beispiele:

```
> Int(2*x*sqrt(1+x^2), x);
```

$$\int 2\,x\,\sqrt{1+x^2}\,dx$$

```
> changevar(1+x^2=u, %, u);
```
$$\int \sqrt{u}\, du$$
```
> value(%);
```
$$\frac{2}{3}\, u^{(3/2)}$$

Als Stammfunktion ergibt sich damit

```
> subs(u=1+x^2, %);
```
$$\frac{2}{3}\,(1+x^2)^{(3/2)}$$
```
> Int(2*(1+x^2)*x, x=2..3);
```
$$\int_2^3 2\,(1+x^2)\,x\,dx$$
```
> changevar(u=x^2+1, %, u);
```
$$\int_5^{10} u\, du$$
```
> value(%);
```
$$\frac{75}{2}$$

Die partielle Integration (**intparts**) basiert auf der Produktregel. Für integrierbare Funktionen f und g gilt

```
> h := 'h':
> eq := h = f*g;
```
$$eq := h = f\,g$$
```
> D(eq)(x);
```
$$\mathrm{D}(h)(x) = \mathrm{D}(f)(x)\,\mathrm{g}(x) + \mathrm{f}(x)\,\mathrm{D}(g)(x)$$
```
> expand(map(Int, %, x));
```
$$\int \mathrm{D}(h)(x)\,dx = \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx + \int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx$$
```
> isolate(%, Int(f(x)*D(g)(x), x));
```
$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = \int \mathrm{D}(h)(x)\,dx - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$
```
> value(%);
```
$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = \mathrm{h}(x) - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$
```
> value(subs(eq, %));
```
$$\int \mathrm{f}(x)\,\mathrm{D}(g)(x)\,dx = (f\,g)(x) - \int \mathrm{D}(f)(x)\,\mathrm{g}(x)\,dx$$

Beispiel:

```
> Int(x^2*exp(x), x);
```
$$\int x^2\, e^x\, dx$$
```
> intparts(%, x^2);
```
$$x^2\, e^x - \int 2\, x\, e^x\, dx$$
```
> intparts(%, x);
```
$$x^2\, e^x - 2\, x\, e^x + \int 2\, e^x\, dx$$
```
> value(%);
```
$$x^2\, e^x - 2\, x\, e^x + 2\, e^x$$
```
> factor(%);
```
$$e^x\,(x^2 - 2\,x + 2)$$

Durch Partialbruchzerlegung (**convert**,**parfrac**) kann man die Integration rationaler Funktionen oft auf die Integration einfacherer Funktionen zurückführen. Es folgt ein einfaches Beispiel:

```
> f := x -> (x^3-3*x+2*x-5)/((x^2-x-1)*(x-3)*(x-4));
```
$$f := x \to \frac{x^3 - x - 5}{(x^2 - x - 1)\,(x-3)\,(x-4)}$$

```
> convert(f(x), parfrac, x);
```

$$-\frac{19}{5}\,\frac{1}{x-3}+5\,\frac{1}{x-4}-\frac{1}{5}\,\frac{2+x}{x^2-x-1}$$

```
> Int(%, x);
```

$$\int -\frac{19}{5}\,\frac{1}{x-3}+5\,\frac{1}{x-4}-\frac{1}{5}\,\frac{2+x}{x^2-x-1}\,dx$$

```
> expand(%);
```

$$-\frac{19}{5}\int\frac{1}{x-3}\,dx+5\int\frac{1}{x-4}\,dx-\frac{2}{5}\int\frac{1}{x^2-x-1}\,dx-\frac{1}{5}\int\frac{x}{x^2-x-1}\,dx$$

```
> value(%);
```

$$-\frac{19}{5}\ln(x-3)+5\ln(x-4)+\frac{1}{5}\sqrt{5}\,\mathrm{arctanh}(\frac{1}{5}\,(2\,x-1)\,\sqrt{5})-\frac{1}{10}\ln(x^2-x-1)$$

Die Probe ergibt

```
> simplify(diff(%, x) - f(x));
```

$$0$$

## 4.3 Algorithmen für die unbestimmte Integration

```
> restart:
```

Die unbestimmte Integration ist eines der "highlights" der Computeralgebra. Neben den klassischen heuristischen Methoden zur Bestimmung von Stammfunktionen, die in vielen Lehrbüchern der Analysis beschrieben sind, benutzt Maple nicht-klassische Integrationsalgorithmen, wie z.B. den Risch-Algorithmus. Im folgenden wird der Algorithmus der Maple-Prozedur int grob beschrieben:

- Zunächst versucht Maple heuristische Methoden zu nutzen: Lookup-tables, Variablensubstitution, Partialbruchzerlegung, partielle Integration, usw..

- Wenn heuristische Methoden nicht zum Erfolg führen, wird der Risch-Algorithmus angewendet, d.h. für elementare (transzendente) Funktionen wird überprüft, ob das unbestimmte Integral als Formel mit endlich vielen Termen, bestehend aus elementaren Funktionen, dargestellt werden kann, und wenn ja, dann wird diese Formel berechnet. Die Klasse der elementaren transzendenten Funktionen umfaßt Polynome, rationale und trigonometrische Funktionen, Exponential- und Logarithmusfunktion sowie alle Funktionen, die durch Verknüpfung und/oder Schachtelung dieser Funktionen entstehen.

- Im letzten Schritt wird die Integration algebraischer Funktionen in RootOf-Notation behandelt (Risch-Trager-Algorithmus). Algebraische Funktionen sind Lösungen von Polynomgleichungen, deren Koeffizienten elementare transzendente Funktionen sein können.

Wenn man wissen möchte, wie Maple zu seinen Ergebnissen kommt, kann man beim Integrieren zusehen. Dazu muß man die Systemvariable **infolevel** für int auf einen Wert zwischen 1 und 5 stellen (je höher der Wert, desto ausführlicher werden die Informationen). Beispiele:

```
> infolevel[int] := 1;
```

$$infolevel_{int} := 1$$

```
> Int(ln(x-1)^2, x): % = value(%);

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/indef2:   applying change of variables

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/ln:   case of integrand containing ln
```

$$\int\ln(x-1)^2\,dx=\ln(x-1)^2\,(x-1)-2\,(x-1)\ln(x-1)+2\,x-2$$

```
> Int((ln(x-1)/x)^2, x): % = value(%);

int/indef1:   first-stage indefinite integration
```

```
int/indef2:   second-stage indefinite integration

int/indef2:   applying change of variables

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/ln:   case of integrand containing ln

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/risch:   enter Risch integration

int/risch/algebraic1:   RootOfs should be algebraic numbers and
functions

int/risch:   exit Risch integration
```

$$\int \frac{\ln(x-1)^2}{x^2}\,dx = \int \frac{\ln(x-1)^2}{x^2}\,dx$$

Maple gibt die Eingabe zurück, d.h. das unbestimmte Integral kann nicht als geschlossene Formel mit endlich vielen elementaren Funktionen ausgedrückt werden. In einigen Fällen wird die Klasse der elementaren Funktionen um neudefinierte (spezielle) Funktionen erweitert. Beispiele (Fehlerfunktion und Exponentialintegral):

```
> infolevel[int] := 0;
```

$$infolevel_{int} := 0$$

```
> Int(exp(-x^2), x) = int(exp(-x^2), x);
```

$$\int e^{(-x^2)}\,dx = \frac{1}{2}\sqrt{\pi}\,\mathrm{erf}(x)$$

```
> Int(exp(-x)/x, x) =int(exp(-x)/x, x);
```

$$\int \frac{e^{(-x)}}{x}\,dx = -\mathrm{Ei}(1,\,x)$$

Integration einer einfachen algebraischen Funktion:

```
> f := x -> 1/(1+x^3)^(1/3);
```

$$f := x \to \frac{1}{(1+x^3)^{(1/3)}}$$

```
> infolevel[int] := 1;
```

$$infolevel_{int} := 1$$

```
> ia := Int(f(x), x);
```

$$ia := \int \frac{1}{(1+x^3)^{(1/3)}}\,dx$$

```
> value(ia);

int/indef1:   first-stage indefinite integration

int/algebraic2/algebraic:   algebraic integration

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/risch:   enter Risch integration

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration
```

```
int/indef2:   trying integration by parts

int/risch:   exit Risch integration
```

$$x\,\mathrm{hypergeom}([\frac{1}{3},\,\frac{1}{3}],\,[\frac{4}{3}],\,-x^3)$$

Das unbestimmte Integral wird hier als hypergeometrische Funktion ${}_2F_1$ zurückgegeben, d.h. als unendliche Reihe. Maple kann dieses Integral jedoch auch als Ausdruck mit endlich vielen elementaren Funktionen darstellen. Die Integration erfolgt dann über den Risch-Trager-Algorithmus. Dieses Verfahren wird aber nur angewendet, wenn der Integrand in der RootOf-Notation erscheint. Die Funktion **RootOf** dient als Platzhalter für die Nullstellen von Gleichungen mit einer Variablen. In Maple ist die RootOf-Notation für algebraische Zahlen und Funktionen voreingestellt. Die Konvertierung in die RootOf-Darstellung erfolgt mit **convert**,**RootOf**

```
> ia1 := convert(ia, RootOf);
```

$$ia1 := \int \frac{1}{\mathrm{RootOf}(\_Z^3 - 1 - x^3)}\,dx$$

Die folgende Integration erfordet viel Rechenzeit und sollte daher nur auf Computern mit schneller CPU durchgeführt werden.

```
> settime := time();
```

$$settime := 313.639$$

```
> ia1 := value(ia1);

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/rischnorm:   enter Risch-Norman integrator

int/rischnorm:   exit Risch-Norman integrator

int/algrisch/int:   Risch/Trager's algorithm for algebraic function

int/algrisch/int:   computation of the algebraic part: start time
315.560

int/algrisch/int:   computation of the algebraic part: end time
315.600

int/algrisch/int:   computation of the transcendental part: start time
  315.680

int/algrisch/int:   computation of the transcendental part: end time
453.259
```

$$ia1 := -\frac{1}{3}\ln(\%1 + \%1\,x^3 - \%1^2\,x^3 + 1 + 2\,x^3 - 3\,\%2\,x^2 + 3\,\%2\,\%1\,x^2 - 3\,\%2^2\,\%1\,x) + \frac{1}{3}$$
$$\%1\ln(2 + \%1\,x^3 + 2\,x^3 - \%1 - \%1^2\,x^3 + 3\,\%2\,\%1\,x^2 - 3\,\%2\,x^2 - 3\,\%2^2\,\%1\,x)$$
$$\%1 := \mathrm{RootOf}(\_Z^2 - \_Z + 1)$$
$$\%2 := \mathrm{RootOf}(\_Z^3 - 1 - x^3)$$

```
> time_consumed := time() - settime;
```

$$time\_consumed := 139.740$$

**convert**,**radical** liefert schließlich die Wurzeldarstellung:

```
> convert(ia1, radical);
```

$$-\frac{1}{3}\ln(\frac{3}{2} - \frac{1}{2}\,I\,\sqrt{3} + \%1\,x^3 - \%1^2\,x^3 + 2\,x^3 - 3\,(1+x^3)^{(1/3)}\,x^2 + 3\,(1+x^3)^{(1/3)}\,\%1\,x^2$$
$$- 3\,(1+x^3)^{(2/3)}\,\%1\,x) + \frac{1}{3}\%1\ln(\frac{3}{2} + \%1\,x^3 + 2\,x^3 + \frac{1}{2}\,I\,\sqrt{3} - \%1^2\,x^3$$
$$+ 3\,(1+x^3)^{(1/3)}\,\%1\,x^2 - 3\,(1+x^3)^{(1/3)}\,x^2 - 3\,(1+x^3)^{(2/3)}\,\%1\,x)$$
$$\%1 := \frac{1}{2} - \frac{1}{2}\,I\,\sqrt{3}$$

Probe:

```
> simplify(diff(%, x) -f(x));
```

$$0$$

## 4.4 Bestimmte Integration

```
> restart:
```

Für die bestimmte Integration wird in der Prozedur int nicht einfach das entsprechende unbestimmte Integral berechnet und dann an den Integrationsgrenzen ausgewertet. Dies würde oft zu Fehlern führen, wie im folgenden Beispiel:

```
> Int(1/x^2, x) = int(1/x^2, x);
```

$$\int \frac{1}{x^2}\,dx = -\frac{1}{x}$$

```
> Int(1/x^2, x=-1..1) = subs(x=1, rhs(%)) - subs(x=-1, rhs(%));
```

$$\int_{-1}^{1} \frac{1}{x^2}\,dx = -2$$

Der Integrand hat im Intervall [-1, 1] eine nicht behebbare Singularität an der Stelle x=0. Maple überprüft die Stetigkeit des Integranden im vorgegebenen Integrationsintervall.

```
> Int(1/x^2, x=-1..1) = int(1/x^2, x=-1..1);
```

$$\int_{-1}^{1} \frac{1}{x^2}\,dx = \infty$$

Für die bestimmte Integration werden in der Prozedur int unterschiedliche Techniken genutzt, wie z.B. lookup-tables, pattern matching, Differentiation von speziellen Funktionen nach Parametern, u.a..

```
> infolevel[int]:=2;
```

$$infolevel_{int} := 2$$

```
> i1 := Int(exp(-u*x^2), x=0..infinity): % = value(%);

Definite integration: Can't determine if the integral is convergent.

Need to know the sign of --> u

Will now try indefinite integration and then take limits.

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/exp:   case of integrand containing exp

int/indef1:   first-stage indefinite integration

int/indef2:   second-stage indefinite integration

int/exp:   case of integrand containing exp

int/prpexp:   case ratpoly*exp(arg)
```

$$\int_0^{\infty} e^{(-u\,x^2)}\,dx = \lim_{x\to\infty} \frac{1}{2}\,\frac{\sqrt{\pi}\,\mathrm{erf}(\sqrt{u}\,x)}{\sqrt{u}}$$

In diesem Beispiel hängt das bestimmte Integral von einem Parameter ab. Die Existenz des Integrals wird bestimmt durch das Vorzeichen von u. Nach der Festlegung der Parametereigenschaft mit **assume**

```
> assume(u>0);
```

wird das Integral ausgewertet.

```
> value(i1);
```

$$\frac{1}{2}\,\frac{\sqrt{\pi}}{\sqrt{u\tilde{}}}$$

```
> about(u);

Originally u, renamed u~:
```

```
is assumed to be: RealRange(Open(0),infinity)
```

Variablen mit definierten Eigenschaften erhalten in Maple als Kennzeichen eine Tilde. Die Parametereigenschaft wird durch die Zuweisung

```
> u := 'u':
```

wieder gelöscht.

```
> about(u);

u:

  nothing known about this object
```

```
> i2 := Int(exp(-u*x^2)*erf(x), x=0..infinity);
```

$$i2 := \int_0^{\infty} e^{(-u\,x^2)}\,\mathrm{erf}(x)\,dx$$

```
> value(i2);
```

$$\frac{\mathrm{hypergeom}([\frac{1}{2},\,1],\,[\frac{3}{2}],\,-\frac{1}{u})}{\sqrt{\pi}\,u}$$

Dieses bestimmte Integral wird von Maple wieder als spezielle hypergeometrische Funktion ${}_2F_1$,
d.h. als unendliche Reihe dargestellt. Die hypergeometrische Funktion läßt sich in eine sehr einfache elementare Funktion konvertieren:

```
> convert(%, StandardFunctions);
```

$$\frac{\arctan(\frac{1}{\sqrt{u}})}{\sqrt{\pi}\,\sqrt{u}}$$

Die Integration liefert dieses Ergebins direkt, wenn wir für den Parameter u

```
> assume(u>0);
```

annehmen:

```
> value(i2);
```

$$\frac{\arctan(\frac{1}{\sqrt{\tilde{u}}})}{\sqrt{\pi}\,\sqrt{\tilde{u}}}$$

## 4.5 Numerische Integration

```
> restart:
```

Für die numerische Berechnung bestimmter Integrale stehen im Paket student mehrere Verfahren zur Verfügung.

```
> with(student):
```

Die Rechteckregeln haben wir bereits bei der Einführung des Riemann-Integrals kennengelernt. Für die numerische Approximation eines Integrals mittels "linksseitiger" Rechtecke gilt

```
> Int(f(x), x=a..b) = limit(leftsum(f(x), x=a..b, n), n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty} \frac{(b-a)\,\left(\sum_{i=0}^{n-1} \mathrm{f}(a+\frac{i\,(b-a)}{n})\right)}{n}$$

Werden die Rechtecke durch Trapeze ersetzt, so erhalten wir die Trapezregel (**trapezoid**).

```
> Int(f(x), x=a..b) = limit(trapezoid(f(x), x=a..b, n), n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty} \frac{1}{2}\,\frac{(b-a)\,\left(\mathrm{f}(a)+2\,\left(\sum_{i=1}^{n-1} \mathrm{f}(a+\frac{i\,(b-a)}{n})\right)+\mathrm{f}(b)\right)}{n}$$

Theoretisch läßt sich zeigen, daß der Approximationsfehler kleiner als

```
> M*(b-a)^3/(12*n^2);
```

$$\frac{1}{12}\,\frac{M\,(b-a)^3}{n^2}$$

ist. Hierbei ist M der maximale Wert der zweiten Ableitung von f auf [a, b]. Wird die Kurve der Funktion f stückweise durch quadratische Polynome q ersetzt mit der Eigenschaft, daß die Funktionswerte von f und q jeweils an den Intervallgrenzen und in der Intervallmitte übereinstimmen, so folgt die Simpson-Regel (**simpson**).

```
> Int(f(x), x=a..b) = limit(simpson(f(x), x=a..b, n), n=infinity);
```

$$\int_a^b \mathrm{f}(x)\,dx = \lim_{n\to\infty}\frac{1}{3}\,\frac{(b-a)\left(\mathrm{f}(a)+\mathrm{f}(b)+4\left(\sum_{i=1}^{1/2\,n}\mathrm{f}(a+\frac{(2\,i-1)\,(b-a)}{n})\right)+2\left(\sum_{i=1}^{1/2\,n-1}\mathrm{f}(a+2\,\frac{i\,(b-a)}{n})\right)\right)}{n}$$

Der Approximationsfehler der Simpson-Regel wird durch

```
> M*(b-a)^5/(180*n^4);
```

$$\frac{1}{180}\,\frac{M\,(b-a)^5}{n^4}$$

begrenzt. M ist hier der maximale Wert der vierten Ableitung von f auf [a,b]. Beispiel:

```
> f := x -> sin(x);
```

$$f := \sin$$

```
> Int(f(x), x=0..Pi/2) = int(f(x), x=0..Pi/2);
```

$$\int_0^{1/2\,\pi}\sin(x)\,dx = 1$$

```
> simplify(value(leftsum(f(x), x=0..Pi/2, n)));
```

$$-\frac{1}{4}\,\frac{\pi\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1+\sin(\frac{1}{2}\,\frac{\pi}{n}))}{n\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1)}$$

```
> seq(evalf(%), n=1..8);
```

0, .5553603666, .7152492302, .7907662606, .8346821305, .8633821947, .8836004770, .8986104019

```
> simplify(value(trapezoid(f(x), x=0..Pi/2, n)));
```

$$-\frac{1}{4}\,\frac{\pi\sin(\frac{1}{2}\,\frac{\pi}{n})}{n\,(\cos(\frac{1}{2}\,\frac{\pi}{n})-1)}$$

```
> seq(evalf(%), n=1..8);
```

.7853981635, .9480594481, .9770486184, .9871158016, .9917617635, .9942818890, .9958002145, .9967851722

```
> combine(expand(value(simpson(f(x), x=0..Pi/2, n))),trig);
```

$$\frac{-4\,\pi\sin(\frac{1}{2}\,\frac{\pi}{n})-\pi\sin(\frac{\pi}{n})}{6\,n\cos(\frac{\pi}{n})-6\,n}$$

```
> seq(evalf(%), n=1..8);
```

1.047197551, 1.002279878, 1.000431595, 1.000134584, 1.000054761, 1.000026313, 1.000014172, 1.000008296

Die Maple-Prozedur int bietet auch numerische Integrationsverfahren (**int**,**numeric**).

```
> int(exp(arcsin(x)), x=0..1);
```

$$\int_0^1 e^{\arcsin(x)}\,dx$$

```
> evalf(%);
```

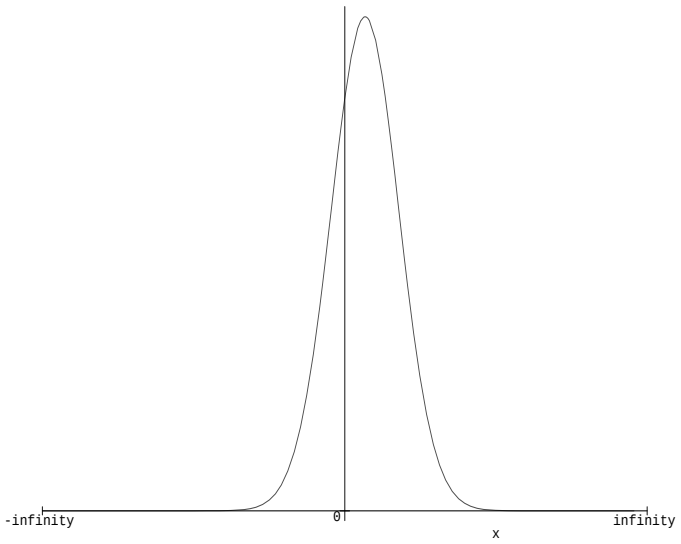
$$1.905238690$$

Die Clenshaw-Curtis-Quadratur wird in der Prozedur int als Standardverfahren angewandt. Zur Behandlung von Singularitäten werden symbolische Methoden wie Variablentransformation und verallgemeinerte Reihenentwicklung eingesetzt.

```
> g := x -> exp(x-x^2/2)/(1+1/2*exp(x));
```

$$g := x \to \frac{e^{(x-1/2\,x^2)}}{1+\frac{1}{2}\,e^x}$$

```
> plot(g(x), x = -infinity..infinity);
```




```
> evalf(int(g(x), x = -infinity..infinity));
```

$$\sum_{\_k1=0}^{\infty} ((-1)^{\_k1}\,2^{(\_k1+1/2)}(\pi\,\mathrm{hypergeom}([],\,[],\,\frac{1}{2}\,(\_k1+1)^2\,(1-\frac{1}{\_k1+1})^2)$$
$$-\sqrt{\pi}\,(\_k1+1)\,(1-\frac{1}{\_k1+1})\,\sqrt{2}\,\mathrm{hypergeom}([1],\,[\frac{3}{2}],\,\frac{1}{2}\,(\_k1+1)^2\,(1-\frac{1}{\_k1+1})^2)$$
$$)\Big/\sqrt{\pi}+2^{(-1/2-\_k1)}\,(-1)^{\_k1}\,(\pi\,\mathrm{hypergeom}([],\,[],\,\frac{1}{2}\,(\_k1+1)^2)$$
$$-\sqrt{\pi}\,(\_k1+1)\,\sqrt{2}\,\mathrm{hypergeom}([1],\,[\frac{3}{2}],\,\frac{1}{2}\,(\_k1+1)^2))\Big/\sqrt{\pi})$$

Das bestimmte Integral wird zunächst formal als unendliche Reihe dargestellt, die dann aber nicht mehr numerisch ausgewertet werden kann. Eine direkte numerische Integration erfolgt, wenn wir int durch die träge Form Int ersetzen:

```
> evalf(Int(g(x), x = -infinity..infinity),15);
```

$$1.80557706229705$$

Alternative Quadraturverfahren (Newton-Cotes) können über den vierten Parameter der Prozedur Int angewählt werden.

```
> evalf(Int(g(x), x = -infinity..infinity, 15, _NCrule));
```

$$1.80557706229705$$

# Kapitel 5

# Programmieren in Maple

## 5.1 Verzweigungen, Schleifen

```
> restart:
```

Maple besitzt eine prozedurale Programmiersprache, deren Aufbau den Programmiersprachen wie Pascal, C oder Fortran sehr ähnlich ist. Maple kennt acht verschiedene Anweisungen (**statements**):

- assignment statement
- selection statement
- repetition statement
- read statement
- save statement
- empty statement
- quit statement
- expressions

Die Wertzuweisung (**assignment** statement) hatten wir bereits kennengelernt. Wir kommen jetzt zur Syntax von **if** Abfragen (selection statement). Verzweigungen, Fallunterscheidungen und Abfragen werden in Maple mit den Schlüsselwörtern if, then, elif, else und fi gebildet. Es gibt vier verschiedene Abfrageformen. Es folgt die Syntax der ersten zwei Formen:

if <expr> then <statseq> fi;

if <expr> then <statseq1> else <statseq2> fi;

Hierbei ist <expr> ein Boolscher Ausdruck oder eine Boolsche Funktion und <statseq> eine Folge von Befehlen (statement sequence). Beispiele:

```
> x := -2;
```

$$x := -2$$

```
> if x > 0 then x := x - 1 fi;
> x;
```

$$-2$$

```
> if x < 0 then 0 else 1 fi;
```

$$0$$

Geschachtelte Abfragen sind ebenfalls möglich.

```
> printlevel := 2;
```

$$printlevel := 2$$

Um den Wert einer geschachtelten if Abfrage anzuzeigen, erhöhen wir die globale Variable **printlevel** auf den Wert 2.

```
> if x>1 then 1
> else if x=0 then 0 else -1 fi
> fi;
```

$$-1$$

Für viele Fallunterscheidungen bietet Maple die beiden folgenden Abfrageformen:

if <expr1> then <statseq1> elif <expr2> then <statseq2> fi;

bzw.

if <expr1> then <statseq1> elif <expr2> then <statseq2>
else <statseq3> fi;

Das Konstrukt 'elif ... then ...' kann mehrmals erscheinen. Die Vorzeichenfunktion läßt sich in dieser Abfrageform wie folgt realisieren:

```
> if x < 0 then -1
> elif x = 0 then 0
> else 1
> fi;
```

$$-1$$

Diese Form der if Abfrage kann man auch als case statement ansehen. Im folgenden Beispiel soll der Parameter n nur die vier Werte 0, 1, 2, 3 annehmen:

```
> n := 5;
```

$$n := 5$$

```
> if n = 0 then 0
> elif n = 1 then 1/2
> elif n = 2 then sqrt(2)/2
> elif n = 3 then sqrt(3)/2
> else ERROR(`bad argument`, n)
> fi;

Error, bad argument, 5
```

Wir kommen nun zum **repetition** statement. Als allgemeinste Form kennt Maple die for Schleife. Es gibt zwei Formen: for-from und for-in. Die Syntax der Schleifenform for-from ist wie folgt:

for <name> from <start> by <step> to <finish> while <expr>

do <statseq> od;

Hierbei ist <start>, <step> und <finish> vom Datentyp numeric, d.h. integer, fraction oder float, und <expr> ein Boolscher Ausdruck. Die Schlüsselwörter for, from, by, to und while können fehlen, <statseq> kann ebenfalls fehlen. Für <name>, <start>, <step>, <finish> und <expr> sind die folgenden Werte vordefiniert:

name dummy variable
start 1
step 1
finish infinity
expr true

Beispiele:

```
> for i from 2 to 5
> do i^2 od;
```

$$4$$
$$9$$
$$16$$
$$25$$

```
> i;
```

$$6$$

Nach Beendigung der Schleife hat der Laufindex i den Wert 6 (i + 1). Voll ausgeschrieben hat dieses Beispiel somit die Kontrollstruktur

```
> for i from 2 by 1 to 5 while true
> do i^2 od;
```

$$4$$
$$9$$
$$16$$
$$25$$

Ist der Wert der Schrittweite <step> negativ, so wird im Schleifenindex rückwärts gezählt.

```
> for i from 5 by -1 to 2
> do i^2 od;
```

$$25$$
$$16$$
$$9$$
$$4$$

Die erste Primzahl größer als 10^7 erhält man z.B. mit der Schleife

```
> for i from 10^7 while not isprime(i)
> do od;
> i;
```

$$10000019$$

Der Schleifenkörper ist in diesem Beispiel leer! Werden alle Schleifenbedingungen weggelassen, dann entsteht eine unendliche Schleife.

do <statseq> od;

Die **while** Schleife hat die folgende Syntax:

while <expr> do <statseq> od;

```
> x := 256:
> while x > 1
> do x := x/4 od;
```

$$x := 64$$
$$x := 16$$
$$x := 4$$
$$x := 1$$

Jetzt kommen wir zur Schleifenform for-in.

for <name> in <expr1> while <expr2> do <statseq> od;

Die Scheife durchläuft die Liste der Operanden op(<expr1>) des Ausdrucks <expr1>; <expr2> ist ein Boolscher Ausdruck. Beispiel:

```
> s := sin(y) + ln(y) + y^2;
```

$$s := \sin(y) + \ln(y) + y^2$$

```
> op(s);
```
$$\sin(y),\ \ln(y),\ y^2$$
```
> printlevel := 1:
> for term in s while type(term, function)
> do diff(term, y) od;
```
$$\cos(y)$$
$$\frac{1}{y}$$
```
> type(y^2, function);
```
$$false$$
```
> term;
```
$$y^2$$

In Maple gibt es zwei weitere Kontrollstrukturen für Schleifen: **break** und **next**. Mit break kann man eine Schleife vorzeitig verlassen. Mit next kann man die Anweisungen bis zum Beginn der nächsten Iteration überspringen.

Beispiel:

```
> for i from 3 by 2 do
> if isprime(2^i-1) then print(2^i-1, `ist Primzahl`)
> else break
> fi
> od;
```
$$7,\ ist\ Primzahl$$
$$31,\ ist\ Primzahl$$
$$127,\ ist\ Primzahl$$

Bemerkung: In vielen Fällen lassen sich for Schleifen durch effizientere Kommandos ersetzen, z.B. seq, map, zip.

## 5.2 Prozeduren

```
> restart:
```

In Maple werden Prozeduren mit der proc ... end Syntax geschrieben. Die allgemeine Form einer Prozedur (**procedure**) hat die folgende Gestalt:

proc(<parameter_sequence>)
local <variable_sequence>;
global <variable_sequence>;
options <option_sequence>;
description <string>;
<statement_sequence>
end;

Bis auf den Prozedurrahmen proc() end können alle anderen Teile fehlen. Es folgt ein einfaches Beispiel:

```
> proc(x, y) x^2+y^2 end;
```
$$\mathbf{proc}(x,\ y)\ x^2+y^2\ \mathbf{end}$$

Wie jedem anderen Objekt in Maple kann man der Prozedur auch einen Namen zuweisen.

```
> F := proc(x, y) x^2+y^2 end;
```
$$F := \mathbf{proc}(x,\ y)\ x^2+y^2\ \mathbf{end}$$

Die Prozedur wird durch den Aufruf

```
> F(diff(x^2, x), 3);
```
$$4\,x^2+9$$

ausgeführt. Alle aktuellen Parameter werden zunächst ausgewertet (call by value), anschließend wird jeder formale Parameter durch den entsprechenden aktuellen Parameter ersetzt (call by name). Das Resultat einer Prozedur ist normalerweise der Wert des zuletzt ausgewerteten Befehls. Die Anzahl der aktuellen Parameter muß nicht mit der Anzahl der formalen Parameter übereinstimmen. Überflüssige aktuelle Parameter werden bei der Prozedurausführung ignoriert. Werden beim Aufruf der Prozedur zuwenig aktuelle Parameter angegeben, dann wird nur dann ein Fehler gemeldet, wenn ein Parameter bei der Auswertung des Prozedurkörpers fehlt.

```
> f := proc(x, y ,z)
> if x>y then x else z fi
> end:
> f(1, 2, 3, 4);
```

$$3$$

```
> f(1, 2);

Error, (in f) f uses a 3rd argument, z, which is missing
> f(2, 1);
```

$$2$$

Einfache Prozeduren, bestehend aus einem Ausdruck oder einer if Abfrage, lassen sich alternativ mit Hilfe der Pfeil-Notation als Funktion (**operators**,**functional**) definieren.

```
> G := (x, y) -> x^2+y^2;
```

$$G := (x,\, y) \to x^2 + y^2$$

Bei Funktionen mit einem Argument können die runden Klammern für die Parameterliste fehlen.

```
> H := n -> if n<0 then 0 else 1 fi;
```

$$H := \mathbf{proc}(n)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ n < 0\ \mathbf{then}\ 0\ \ \mathbf{else}\ 1\ \mathbf{fi}\ \mathbf{end}$$

Prozeduren sind gültige Maple-Ausdrücke. Sie können auch ohne Bezeichner als anonyme Prozeduren (Funktionen) verwendet werden.

```
> (x -> x^2)(t);
```

$$t^2$$

```
> proc(x, y) x^2+y^2 end (u, v);
```

$$u^2 + v^2$$

```
> map(x->x^2, [1,2,3,4]);
```

$$[1,\, 4,\, 9,\, 16]$$

```
> D(x -> x^2);
```

$$x \to 2\,x$$

```
> D(exp + 2*ln);
```

$$\exp + 2\,(a \to \frac{1}{a})$$

Die Deklarierung formaler Parameter ermöglicht die Überprüfung von Datentypen. Die Syntax im Prozedurkopf ist

<parameter>::<type>

Beispiele:

```
> MAX := proc(x::numeric, y::numeric)
> if x>y then x else y fi
> end:
> MAX(Pi, 3);

Error, MAX expects its 1st argument, x, to be of type numeric,
but received Pi
> G := proc(n::even)
> n!*(n/2)!
> end:
> G(6);
```

$$4320$$

```
> G(5);

Error, G expects its 1st argument, n, to be of type even,
but received 5
```

Innerhalb einer Prozedur lassen sich mit **nargs** die Anzahl der aktuellen Parameter, mit **args** die Folge der aktuellen Parameter und mit args[i] der i-te aktuelle Parameter ermitteln. Eine allgemeine Prozedur MAX, die das Maximum einer endlichen Zahlenfolge errechnet, läßt sich damit wie folgt realisieren:

```
> MAX := proc()
> local i, m;
> if nargs = 0 then RETURN(FAIL) fi;
> m := args[1];
> for i from 2 to nargs do
> if args[i] > m then m := args[i] fi
> od;
> m
> end:
> MAX(1/2, 4/7, 2/3);
```

$$\frac{2}{3}$$

Variablen innerhalb einer Prozedur sind entweder lokal oder global zu dieser Prozedur. i und m sind in der Prozedur MAX als lokale Variablen vereinbart. Werden Variablen in einer Prozedur nicht als lokal oder global vereinbart, so entscheidet Maple darüber, wie sie behandelt werden. Eine nicht-vereinbarte Variable wird als lokale Variable verwendet, wenn

- sie auf der linken Seite einer Zuweisung steht oder

- als Laufvariable in einer for Schleife steht oder

- als Indexvariable in den Funktionen seq, add oder mul vorkommt.

Andernfalls wird die Variable als globale Variable behandelt.

```
> MAX := proc()
> if nargs = 0 then RETURN(FAIL) fi;
> m := args[1];
> for i from 2 to nargs do
> if args[i] > m then m := args[i] fi
> od;
> m
> end:

Warning, `m` is implicitly declared local

Warning, `i` is implicitly declared local
```

In einer Prozedur werden die lokalen Variablen nur einmal ausgewertet, während globale Variablen beliebig oft ausgewertet werden. Beispiel:

```
> f := a + b;
```

$$f := a + b$$

```
> a := c^2/b;
```

$$a := \frac{c^2}{b}$$

```
> c := b^3 +3;
```

$$c := b^3 + 3$$

Die volle rekursive Auswertung liefert nun

```
> f;
```

$$\frac{(b^3+3)^2}{b} + b$$

Die Auswertungsstufen kann man mit eval kontrollieren:

```
> eval(f, 1);
```

$$a + b$$

```
> eval(f, 2);
```

$$\frac{c^2}{b} + b$$

```
> eval(f, 3);
```

$$\frac{(b^3+3)^2}{b} + b$$

Mit dem eval Kommando läßt sich die volle Auswertung von lokalen Variablen erzwingen.

```
> F := proc()
> local x, y, z;
> x := y^2; y := z; z := 3; eval(x);
> end:
> F();
```

$$9$$

Mit **RETURN**(<sequence>); kann man eine Prozedur an einer vorgewählten Stelle verlassen. Der Wert der Prozedur ist dann <sequence>.

```
> GCD := proc(a::integer, b::integer)
> local g;
> if a=0 and b=0 then RETURN(0,0,0) fi;
> g := igcd(a,b);
> g, iquo(a, g), iquo(b, g);
> end:
> GCD(0, 0);
```

$$0, 0, 0$$

```
> GCD(12, 8);
```

$$4, 3, 2$$

Mit **ERROR**(<sequence>); wird die Abarbeitung der Prozedur unterbrochen und es wird eine Fehlermeldung der From

Error, (in <procname>) <sequence>

ausgegeben.

```
> inv := proc(x::numeric)
> if x = 0 then ERROR("Die Eingabe", x, "liefert Division durch 0");
> else 1/x
> fi;
> end:
> inv(0);

Error, (in inv) Die Eingabe, 0, liefert Division durch 0
```

In bestimmten Situationen ist es sinnvoll, den unausgewerteten Prozeduraufruf zurückzugeben. Beispiel:

```
> MAX := proc(x, y)
> if x > y then x else y fi;
> end:
> MAX(3.2, 2);
```

$$3.2$$

```
> MAX(x, 1/x);

Error, (in MAX) cannot evaluate boolean
```

Die Prozedur MAX kann natürlich keine symbolischen Parameter verarbeiten. Beim Plotten der Funktion MAX tritt die gleiche Fehlersituation auf.

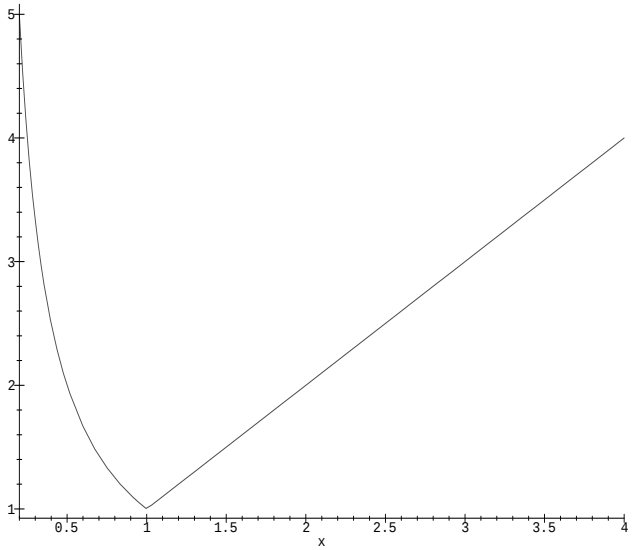
```
> plot(MAX(x, 1/x), x=1/2..2);

Error, (in MAX) cannot evaluate boolean
```

Maple wertet hier zunächst den Ausdruck MAX(x, 1/x) symbolisch aus, bevor das plot Kommando aufgerufen wird. Eine Lösung des Problems ist hier, die Prozedur zu erweitern und bei symbolischen Parametern den Aufruf unausgewertet zurückzugeben.

```
> MAX := proc(x, y)
> if type(x, numeric) and type(y, numeric) then
> if x > y then x else y fi
> else RETURN('procname'(args))
> fi;
> end:
> MAX(x, 1/2);
```

$$\mathrm{MAX}(x, \frac{1}{2})$$

```
> plot(MAX(x, 1/x), x=0.2..4);
```




Manchmal möchte man, daß Werte über Prozedurparameter zurückgegeben werden. Die folgende Prozedur MEMBER bestimmt, ob eine gegebene Liste L einen Ausdruck x enthält. Darüberhinaus soll die Position von x in L über den dritten Parameter p zurückgegeben werden.

```
> MEMBER := proc(x::anything, L::list, p::name)
> local i;
> for i to nops(L) do
> if x = L[i] then
> if nargs > 2 then p := i fi;
> RETURN(true)
> fi;
> od;
> false
> end:
> MEMBER(b, [a, b, c, d], q);
```

$$true$$

```
> q;
```

$$2$$

## 5.3 Optionen und interne Prozedurverwaltung

```
> restart:
```

Eine Prozedur kann eine oder mehrere Optionen (**options**) haben. Mögliche Optionen sind: **remember**, builtin, system, arrow, angle, trace und Copyright. Die Option remember ist für Anwendungen von besonderer Bedeutung. Wird eine Prozedur mit der Option remember aufgerufen, so erfolgt ein Eintrag in die remember **table** dieser Prozedur. Dabei werden die aktuellen Eingabeparameter (**indices**) eindeutig mit dem Resultat der Prozedurauswertung (**entries**) verknüpft. Maple überprüft bei jedem Prozeduraufruf, ob ein Eintrag in der remember table mit den aktuellen Parametern übereinstimmt. Ist dies der Fall, dann wird die Prozedur nicht durchlaufen, sondern der Eintrag aus der remember table wird ausgegeben. Im folgenden Beispiel werden die Tschebyscheff-Polynome mit Hilfe der Rekursionsbeziehung T(n, x) = 2xT(n-1, x) - T(n-2, x), n $>$ 1, berechnet.

```
> T := proc(n, x)
> option remember;
> 2*x*T(n-1, x) - T(n-2, x)
> end;
```

$$T := \mathbf{proc}(n,\, x)\ \mathbf{option}\ \mathit{remember};\ 2 \times x \times \mathrm{T}(n-1,\, x) - \mathrm{T}(n-2,\, x)\ \mathbf{end}$$

Wir definieren die beiden Startwerte der Rekursion explizit. Die Zuweisungen werden dann automatisch in die remember table eingetragen. Diese Methode haben wird schon bei stetig ergänzbaren Funktionen angewandt; sie funktioniert auch ohne remember option.

```
> T(0, x) := 1:
> T(1, x) := x:
```

Die remember table ist der vierte Operand einer Maple-Prozedur.

```
> op(4, eval(T));
```

$$\begin{array}{l}\mathrm{table}([\\ \quad (0,\, x) = 1\\ \quad (1,\, x) = x\\ \quad ])\end{array}$$

Bei jedem Prozeduraufruf mit neuen Eingabeparametern werden weitere Einträge in die remember table aufgenommen.

```
> T(4, x);
```

$$2\,x\,(2\,x\,(2\,x^2-1)-x)-2\,x^2+1$$

```
> op(4, eval(T));
```

$$\begin{array}{l}\mathrm{table}([\\ \quad (4,\, x) = 2\,x\,(2\,x\,(2\,x^2-1)-x)-2\,x^2+1\\ \quad (3,\, x) = 2\,x\,(2\,x^2-1)-x\\ \quad (0,\, x) = 1\\ \quad (2,\, x) = 2\,x^2-1\\ \quad (1,\, x) = x\\ \quad ])\end{array}$$

Mit der Funktion **forget** kann man einzelne Einträge oder auch die gesamte remember table löschen.

```
> readlib(forget):
> forget(T, 4, x); # Bug in Release 5: forget löscht alle Einträge
> op(4, eval(T));
> forget(T);
> op(4, eval(T));
```

Prozeduren werden in Pfeil-Notation ausgegeben, wenn die Optionen operator und arrow eingeschaltet werden.

```
> f := proc(x)
> option operator, arrow;
> x^2
> end;
```

$$f := x \to x^2$$

Jede Option, die mit dem Wort Copyright anfängt, wird als Copyright Option interpretiert. Bei Copyright geschützten Prozeduren wird der Prozedurkörper nur dann angezeigt, wenn die interface Variable verboseproc den Wert 2 hat.

```
> d := proc(expr::anything, x::name)
> option `Copyright (c) 1684 by G.W. Leibniz`;
> Diff(expr, x);
> end;
```

$$d := \mathbf{proc}(expr{::}anything,\ x{::}name)\ \ldots\ \mathbf{end}$$

```
> d(sin(x), x);
```

$$\frac{\partial}{\partial x}\sin(x)$$

```
> value(%);
```

$$\cos(x)$$

Wir setzen jetzt die interface Variable **verboseproc** auf 2. Der Quelltext einer Prozedur wird dann mit print ausgegeben.

```
> interface(verboseproc=2);
> print(d);
```

$$\begin{array}{l}\mathbf{proc}(expr{::}anything,\ x{::}name)\\ \quad \mathbf{option}\ `Copyright\ (c)\ 1684\ by\ G.W.\ Leibniz`;\\ \quad\quad \mathrm{Diff}(expr,\ x)\\ \quad \mathbf{end}\end{array}$$

Bei Prozeduren aus der Maple-Library läßt sich der Quelltext in analoger Weise anzeigen.

```
> print(ln);
```

$$\begin{array}{l}\mathbf{proc}(x{::}algebraic)\\ \quad \mathbf{option}\ `Copyright\ (c)\ 1992\ by\ the\ University\ of\ Waterloo.\ All\ rights\ reserved.`;\\ \quad\quad \mathbf{if}\ \mathrm{nargs} \neq 1\ \mathbf{then}\ \mathrm{ERROR}(`expecting\ 1\ argument,\ got\ `.\mathrm{nargs})\\ \quad\quad \mathbf{elif}\ \mathrm{type}(x,\ \text{'complex}(\mathit{float})\text{'})\ \mathbf{then}\ \mathrm{evalf}(\text{'ln'}(x))\\ \quad\quad \mathbf{elif}\ x = 0\ \mathbf{then}\ \mathrm{ERROR}(`singularity\ encountered`)\\ \quad\quad \mathbf{elif}\ \mathrm{type}(x,\ \text{'}\mathit{function}\text{'})\ \mathbf{and}\ \mathrm{op}(0,\ x) = \text{'exp'}\ \mathbf{and}\ \Im(\mathrm{op}(1,\ x)) = 0\ \mathbf{then}\ \mathrm{op}(1,\ x)\\ \quad\quad \mathbf{elif}\ \mathrm{type}(x,\ \text{'}\mathit{rational}\text{'})\ \mathbf{and}\ \mathrm{numer}(x) = 1\ \mathbf{and}\ 0 < x\ \mathbf{then}\ -\ln(\mathrm{denom}(x))\\ \quad\quad \mathbf{elif}\ \mathrm{type}(x,\ \text{'}\mathit{constant}^{numeric}\text{'})\ \mathbf{and}\ \mathrm{signum}(\mathrm{op}(1,\ x)) = 1\ \mathbf{then}\\ \quad\quad\quad \mathrm{op}(2,\ x) \times \ln(\mathrm{op}(1,\ x))\\ \quad\quad \mathbf{elif}\mathrm{type}(x,\ \text{'`^`'})\ \mathbf{and}\ (\mathrm{type}(\mathrm{op}(2,\ x),\ \text{'}\mathit{integer}\text{'})\ \mathbf{or}\ \mathrm{is}(\mathrm{op}(2,\ x),\ \text{'}\mathit{integer}\text{'}))\ \mathbf{and}\\ \quad\quad (\mathrm{signum}(0,\ \Re(\mathrm{op}(1,\ x)),\ 1) = 1\ \mathbf{or}\ \mathrm{member}(\mathrm{signum}(0,\ \Im(\mathrm{op}(1,\ x)),\ 0),\ \{-1,\ 1\}))\\ \quad\quad \mathbf{then}\ \mathrm{op}(2,\ x) \times \ln(\mathrm{op}(1,\ x))\\ \quad\quad \mathbf{else}\ \ln(x) := \text{'ln'}(x)\\ \quad\quad \mathbf{fi}\\ \quad \mathbf{end}\end{array}$$

Eine Maple-Prozedur kann bis zu 6 Operanden haben:

- formale Parameter
- lokale Variablen
- Optionen
- remember table
- description
- globale Variablen

Beispiel:

```
> g := proc(x::list, n::posint)
> local i;
> global y;
> option remember, Copyright;
> description `Summenbildung`;
> sum(x[i]+y[i], i=1..n);
> end:
> y := [1, 2, 3];
```

$$y := [1, 2, 3]$$

```
> g([1,2,3], 3);
```

$$12$$

```
> for k to 6 do
> op(k, eval(g));
> od;
```

$$x{::}list,\ n{::}posint$$

$$i$$

$$remember,\ Copyright$$

$$\begin{array}{l}\text{table}([\\ \quad([1,\ 2,\ 3],\ 3) = 12\\ \quad])\end{array}$$

$$Summenbildung$$

$$y$$

Maple erlaubt die Definition geschachtelter Prozeduren, d.h. Prozeduren können innerhalb anderer Prozeduren definiert werden oder von Prozeduren übergeben werden. Lexical scoping ermöglicht es, daß in geschachtelten Prozeduren die Variablen der umgebenen Prozedur genutzt werden können. Mit lexical scoping wird eine bessere Einkapselung von Maple-Programmen erreicht. Mit der folgenden einfachen Maple-Funktion werden spezielle Matrizen generiert. Die Matrixstruktur wird über eine innere anonyme Prozedur definiert, die Dimension der Matrix und ein Parameter der Matrix werden über zwei Eingabeparameter einer äußeren Prozedur gesteuert. Die innere Prozedur soll dabei auf die Eingabeparameter zurückgreifen.

```
> interface(warnlevel=4);
> A := (n,a) -> matrix(n,n,(i,k) -> n+1+a - max(i,k));

Warning, `n` is a lexically scoped parameter

Warning, `a` is a lexically scoped parameter
```

$$A := (n,\ a) \to \text{matrix}(n,\ n,\ (i,\ k) \to n + 1 + a - \max(i,\ k))$$

Die Interface-Variable **warnlevel** ist hier auf 4 gesetzt (voreingestellt ist 3), damit warning Meldungen zur Kompatibilität ausgegeben werden.

```
> A(4,5);
```

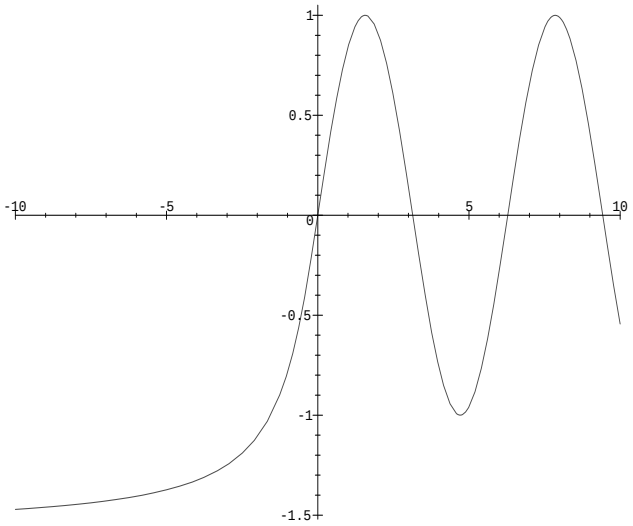
$$\begin{bmatrix} 9 & 8 & 7 & 6 \\ 8 & 8 & 7 & 6 \\ 7 & 7 & 7 & 6 \\ 6 & 6 & 6 & 6 \end{bmatrix}$$

Am Ende dieses Kapitels wollen wir noch ein Beispiel für algorithmisches Differenzieren angeben. Der Differentialoperator D ermöglicht die Differentiation von Maple-Prozeduren (algorithmische Differentiation), d.h. es können z.B. stückweise definierte oder iterativ definierte Funktionen differenziert werden. Beispiel:

```
> S := x -> if x > 0 then sin(x) else arctan(x) fi;
```

$$S := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ 0 < x\ \mathbf{then}\ \sin(x)\ \mathbf{else}\ \arctan(x)\ \mathbf{fi}\ \mathbf{end}$$

```
> plot(S, -10..10);
```




Die Funktion S ist an der Stelle x=0 differenzierbar.

```
> Limit((sin(h) - sin(0))/h, h=0, right) = Limit((arctan(h) - arctan(0))/h,
> h=0, left);
```

$$\lim_{h \to 0+} \frac{\sin(h)}{h} = \lim_{h \to 0-} \frac{\arctan(h)}{h}$$

```
> value(%);
```

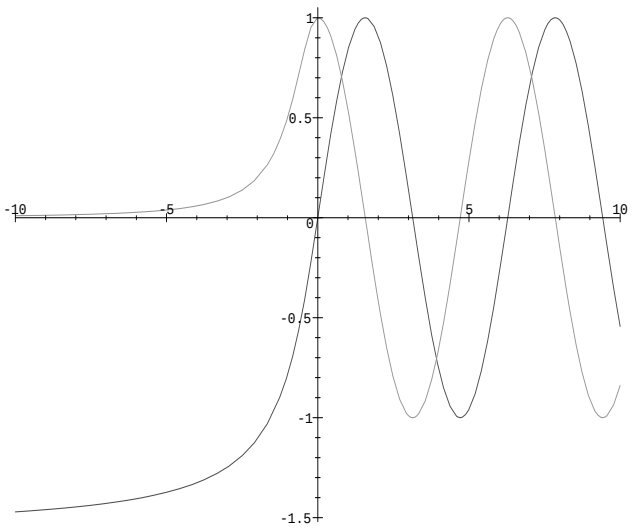
$$1 = 1$$

Als Ableitungsfunktion erhalten wir die Prozedur

```
> Sp := D(S);
```

$$Sp := \mathbf{proc}(x)\ \mathbf{option}\ operator,\ arrow;\ \mathbf{if}\ 0 < x\ \mathbf{then}\ \cos(x)\ \mathbf{else}\ 1 \Big/ (x^2 + 1)\ \mathbf{fi}\ \mathbf{end}$$

```
> plot({S, Sp}, -10..10);
```


1
0.5
-10
-5
0
5
10
-0.5
-1
-1.5

# Kapitel 6

# Differentialgleichungen

## 6.1 Symbolische Lösung

```
> restart:
```

Die symbolische Lösung von Differentialgleichungen ist ein weiterer Schwerpunkt der Computeralgebra. Wir beschränken uns hier auf die Beschreibung der Maple-Prozedur dsolve, die Algorithmen für die analytische Lösung von gewöhnlichen Differentialgleichungen bereitstellt. Eine (im allgemeinen nichtlineare) Gleichung der Form

F(y(t),y'(t),y''(t), ..., y^(n)(t),t) = 0

heißt gewöhnliche Differentialgleichung n-ter Ordnung. In dsolve sind Lösungsalgorithmen für zahlreiche lineare und nichtlineare Differentialgleichungstypen implementiert. Außerdem können lineare Differentialgleichungssysteme gelöst werden. Für die Behandlung nicht lösbarer Differentialgleichungen werden einige numerische Lösungsverfahren angeboten. Die Syntax von **dsolve** ist dsolve(<deqns>, <vars>, <opt>). Hierbei bedeuten <deqns> eine Menge mit Differentialgleichungen und zugehörigen Anfangswerten, <vars> die Menge der gesuchten Lösungsfunktionen und <opt> optionale Argumente. Es folgt eine einfache Differentialgleichung mit Anfangsbedingung:

```
> de1 := diff(v(t), t)+2*t=0;
```

$$de1 := (\frac{\partial}{\partial t}\,\mathrm{v}(t)) + 2\,t = 0$$

```
> ini1 := v(1)=5;
```

$$ini1 := \mathrm{v}(1) = 5$$

```
> dsolve({de1,ini1}, v(t));
```

$$\mathrm{v}(t) = -t^2 + 6$$

Werden die Anfangsbedingungen weggelassen, so gibt dsolve Lösungen mit numerierten Integrationskonstanten _C<n> zurück.

```
> de2 := diff(y(x), x$2) - y(x) = 1;
```

$$de2 := (\frac{\partial^2}{\partial x^2}\,\mathrm{y}(x)) - \mathrm{y}(x) = 1$$

```
> dsolve({de2}, y(x));
```

$$\mathrm{y}(x) = -1 + \_C1\cosh(x) + \_C2\sinh(x)$$

Die Anfangs- oder Randbedingungen für Ableitungen von Funktionen müssen in der folgenden Notation angegeben werden:

D(<function>)(<var_value >)=<value>

und für höhere Ableitungen

(D@@n)(<function>)(<var_value >)=<value>

Um Informationen zur Lösungssuche zu bekommen, erhöhen wir zunächst den Wert der Variablen infolevel.

```
> infolevel[dsolve] := 5;
```

$$infolevel_{dsolve} := 5$$

Beispiele:

```
> de3:=diff(y(t), t$2)+5*diff(y(t), t)+6*y(t)=0;
```

$$de3 := (\frac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + 5\,(\frac{\partial}{\partial t}\,\mathrm{y}(t)) + 6\,\mathrm{y}(t) = 0$$

```
> ini3 := y(0)=0, D(y)(0)=1;
```

$$ini3 := \mathrm{y}(0) = 0,\ \mathrm{D}(y)(0) = 1$$

```
> dsolve({de3,ini3}, y(t));

Methods for second order ODEs:

trying to isolate the derivative d^2y/dx^2

successful isolation of d^2y/dx^2

 -> trying classification methods

trying a quadrature

trying a symmetry of the form [xi=0, eta=F(x)]

dsolve/diffeq/plus_min_expsols:   checking sin-cos solver

dsolve/diffeq/polylinearODE:   trying linear constant coefficient

dsolve/diffeq/polylinearODE:   linear constant coefficient successful
```

$$\mathrm{y}(t) = -e^{(-3\,t)} + e^{(-2\,t)}$$

Manchmal wird die Lösung einer Differentialgleichung in impliziter Form zurückgegeben.

```
> infolevel[dsolve] := 0;
```

$$infolevel_{dsolve} := 0$$

```
> de4 := diff(y(x), x) = exp(y(x))*y(x);
```

$$de4 := \frac{\partial}{\partial x}\,\mathrm{y}(x) = e^{\mathrm{y}(x)}\,\mathrm{y}(x)$$

```
> dsolve(de4, y(x));
```

$$x + \mathrm{Ei}(1,\,\mathrm{y}(x)) + \_C1 = 0$$

Die explizite Lösung kann man (wenn möglich) über die Option explicit angefordern:

```
> dsolve(de4, y(x), explicit);
```

$$\mathrm{y}(x) = \mathrm{RootOf}(x + \mathrm{Ei}(1,\,\_Z) + \_C1)$$

Mit der Option implicit (default) wird die Lösung in impliziter Form angegeben:

```
> dsolve(de4, y(x), implicit);
```

$$x + \mathrm{Ei}(1,\,\mathrm{y}(x)) + \_C1 = 0$$

## 6.2 Laplace-Transformation

```
> restart:
```

Die **Laplace**-Transformation von Differentialgleichungen ermöglicht oft die Reduktion der Komplexität des Problems. Die vorgegebenen Differentialgleichungen werden zunächst mittels Laplace-Transformation in algebraische Gleichungen überführt, die dann mit algebraischen Methoden gelöst werden können. Durch Rücktransformation der algebraischen Lösungen erhält man anschließend die gesuchten Lösungen. Das folgende Beispiel kommt aus der klassischen Mechanik: Zwei Massen, m und $\alpha\,m$ ( $\alpha$ sei ein Faktor), verbunden durch eine Feder, befinden sich auf einer reibungsfreien Fläche. Die Federkonstante sei k. Gesucht sind die Bewegungsbahnen, wenn die erste Masse zur Zeit t=1 angestoßen wird. Die Differentialgleichungen folgen aus dem zweiten Newtonschen Gesetz und dem Hookschen Gesetz für elastische Federn: Kraft = Masse * Beschleunigung = k * Auslenkung

```
> eq1 := alpha*m*diff(x[1](t), t$2) = k*(x[2](t) - x[1](t)) + u(t);
```

$$eq1 := \alpha\,m\,(\frac{\partial^2}{\partial t^2}\,x_1(t)) = k\,(x_2(t) - x_1(t)) + \mathrm{u}(t)$$

```
> eq2 := m*diff(x[2](t), t$2) = k*(x[1](t) - x[2](t));
```

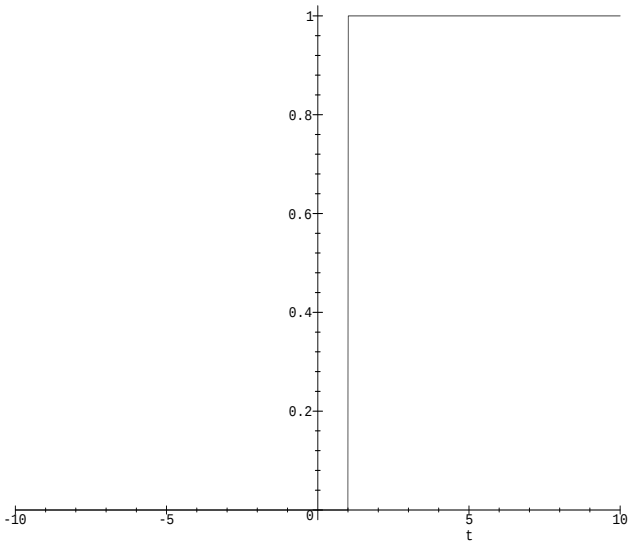
$$eq2 := m\,(\frac{\partial^2}{\partial t^2}\,x_2(t)) = k\,(x_1(t) - x_2(t))$$

Der Stoß zur Zeit t=1 an der ersten Masse werde durch eine Kraft in Form der Einheitssprungfunktion Heaviside beschrieben. Um den Schreibaufwand zu begrenzen und die Ergebnisse etwas übersichtlicher darstellen zu können, wird die Heaviside-Funktion durch $\sigma$ abgekürzt.

```
> alias(sigma=Heaviside):
> u := t -> sigma(t-1);
```

$$u := t \to \sigma(t-1)$$

```
> plot(u(t), t=-10..10);
```




```
> ini := x[1](0) = 2, D(x[1])(0) = 0, x[2](0) = 0, D(x[2])(0) = 0;
```

$$ini := x_1(0) = 2,\ \mathrm{D}(x_1)(0) = 0,\ x_2(0) = 0,\ \mathrm{D}(x_2)(0) = 0$$

```
> # dsolve({eq1,eq2,ini}, {x[1](t),x[2](t)});
```

Die Lösung ohne Laplace-Transformation erfordert viel CPU-Zeit!

```
> dsolve({eq1,eq2,ini}, {x[1](t),x[2](t)}, method=laplace);
```

$$\left\{x_2(t) = 2\,\alpha\,k\left(\frac{1}{k\,(\alpha+1)} - \frac{\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,t)}{k\,(\alpha+1)}\right) + k\left(-\frac{\sigma(t-1)\,\alpha\,m}{k^2\,(\alpha+1)^2} + \frac{1}{2}\,\frac{\sigma(t-1)\,t^2}{k\,(\alpha+1)}\right.\right.$$

$$\left.\left. - \frac{\sigma(t-1)\,t}{k\,(\alpha+1)} + \frac{1}{2}\,\frac{\sigma(t-1)}{k\,(\alpha+1)} + \frac{\sigma(t-1)\,\alpha\,m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,(t-1))}{k^2\,(\alpha+1)^2}\right)\Big/m,\ x_1(t) =$$

$$2\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,t) + 2\,\alpha\,k\left(\frac{1}{k\,(\alpha+1)} - \frac{\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,t)}{k\,(\alpha+1)}\right) + \frac{\sigma(t-1)}{k\,(\alpha+1)}$$

$$- \frac{\sigma(t-1)\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,(t-1))}{k\,(\alpha+1)} + k\left(-\frac{\sigma(t-1)\,\alpha\,m}{k^2\,(\alpha+1)^2} + \frac{1}{2}\,\frac{\sigma(t-1)\,t^2}{k\,(\alpha+1)} - \frac{\sigma(t-1)\,t}{k\,(\alpha+1)}\right.$$

$$+\frac{1}{2}\,\frac{\sigma(t-1)}{k\,(\alpha+1)}+\frac{\sigma(t-1)\,\alpha\,m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\,m}}\,(t-1))}{k^2\,(\alpha+1)^2}\Bigg)\Big/m\Bigg\}$$

Wir erhalten spezielle Lösungsfunktionen, wenn wir die allgemeine Lösung an speziellen Werten für die Konstanten $\alpha$, m und k auswerten:

```
> sol1 := collect(eval(%, {alpha=1/10,m=1,k=1}), sigma(t-1));
```

$$sol1 := \{x_2(t)=(\frac{45}{121}+\frac{5}{11}\,t^2-\frac{10}{11}\,t+\frac{10}{121}\cos(\sqrt{11}\,(t-1)))\,\sigma(t-1)+\frac{2}{11}-\frac{2}{11}\cos(\sqrt{11}\,t),$$
$$x_1(t)=(\frac{155}{121}-\frac{100}{121}\cos(\sqrt{11}\,(t-1))+\frac{5}{11}\,t^2-\frac{10}{11}\,t)\,\sigma(t-1)+\frac{20}{11}\cos(\sqrt{11}\,t)+\frac{2}{11}\}$$

Für die graphische Veranschaulichung definieren wir diese speziellen Lösungen als Maple-Funktionen $y_1$ und $y_2$.

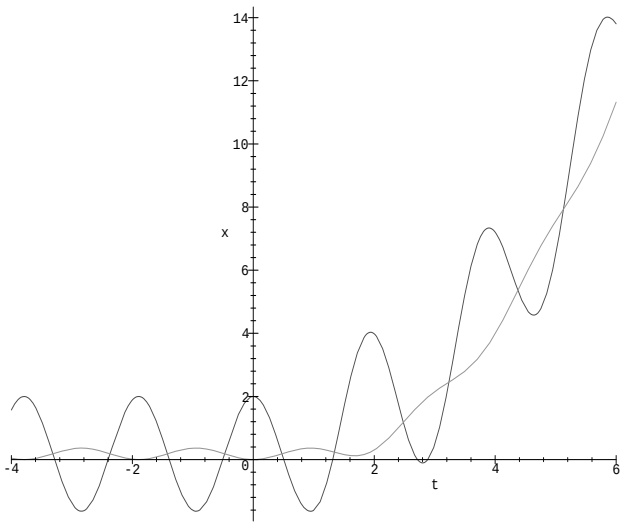
```
> y[1] := unapply(eval(x[1](t), sol1), t);
```

$$y_1 := t\to(\frac{155}{121}-\frac{100}{121}\cos(\sqrt{11}\,(t-1))+\frac{5}{11}\,t^2-\frac{10}{11}\,t)\,\sigma(t-1)+\frac{20}{11}\cos(\sqrt{11}\,t)+\frac{2}{11}$$

```
> y[2] := unapply(eval(x[2](t), sol1), t);
```

$$y_2 := t\to(\frac{45}{121}+\frac{5}{11}\,t^2-\frac{10}{11}\,t+\frac{10}{121}\cos(\sqrt{11}\,(t-1)))\,\sigma(t-1)+\frac{2}{11}-\frac{2}{11}\cos(\sqrt{11}\,t)$$

```
> plot({y[1](t),y[2](t)}, t=-4..6, x);
```




Die Lösung des obigen Differentialgleichungssystems kann man natürlich auch direkt mit Hilfe der Kommandos **laplace** für die Laplace-Transformation und **invlaplace** für die inverse Laplace-Transformation ermitteln.

Die Laplace-Transformation ist eine Integraltransformation, die für (Zeit-)Funktionen f(t) wie folgt erklärt ist:

```
> Int(f(t)*exp(-s*t), t=0..infinity);
```

$$\int_0^\infty \mathrm{f}(t)\,e^{(-s\,t)}\,dt$$

Wir führen nun die folgenden Abkürzungen ein, um die Gleichungen möglichst einfach zu halten:

```
> alias(F1(s)=laplace(x[1](t), t, s)):
> alias(F2(s)=laplace(x[2](t), t, s)):
> with(inttrans);
```

[*addtable*, *fourier*, *fouriercos*, *fouriersin*, *hankel*, *hilbert*, *invfourier*, *invhilbert*, *invlaplace*, *invmellin*, *laplace*, *mellin*, *savetable*]

```
> laplace(eq1, t, s);
```

$$\alpha\, m\,(s\,(s\,\mathrm{F1}(s)-x_1(0))-\mathrm{D}(x_1)(0))=k\,(\mathrm{F2}(s)-\mathrm{F1}(s))+\frac{e^{(-s)}}{s}$$

```
> laplace(eq2, t, s);
```

$$m\,(s\,(s\,\mathrm{F2}(s)-x_2(0))-\mathrm{D}(x_2)(0))=k\,(\mathrm{F1}(s)-\mathrm{F2}(s))$$

```
> eval({%,%%}, {ini});
```

$$\{m\,s^2\,\mathrm{F2}(s)=k\,(\mathrm{F1}(s)-\mathrm{F2}(s)),\ \alpha\, m\, s\,(s\,\mathrm{F1}(s)-2)=k\,(\mathrm{F2}(s)-\mathrm{F1}(s))+\frac{e^{(-s)}}{s}\}$$

Dieses Gleichungssystem in den Laplace-Transformierten F1(s) und F2(s) ist linear und kann leicht mit solve gelöst werden.

```
> solve(%, {F1(s),F2(s)});
```

$$\{\mathrm{F1}(s)=\frac{(2\,\alpha\, m\, s^2\, e^s+1)\,(m\,s^2+k)}{e^s\,s^3\,m\,(\alpha\, m\, s^2+\alpha\, k+k)},\ \mathrm{F2}(s)=\frac{k\,(2\,\alpha\, m\, s^2\, e^s+1)}{e^s\,s^3\,m\,(\alpha\, m\, s^2+\alpha\, k+k)}\}$$

Die inverse Laplace-Transformation liefert schließlich die Lösungsfunktionen des ursprünglichen Differentialgleichungssystems.

```
> invlaplace(%, s, t);
```

$$\left\{x_2(t)=k\left(2\,\frac{\alpha\, m}{k\,(\alpha+1)}-2\,\frac{\alpha\, m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,t)}{k\,(\alpha+1)}-\frac{\sigma(t-1)\,\alpha\, m}{k^2\,(\alpha+1)^2}+\frac{1}{2}\,\frac{\sigma(t-1)\,t^2}{k\,(\alpha+1)}\right.\right.$$

$$\left.-\frac{\sigma(t-1)\,t}{k\,(\alpha+1)}+\frac{1}{2}\,\frac{\sigma(t-1)}{k\,(\alpha+1)}+\frac{\sigma(t-1)\,\alpha\, m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,(t-1))}{k^2\,(\alpha+1)^2}\right)\Big/m,x_1(t)=\Bigg($$

$$2\,m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,t)+2\,\frac{\alpha\, m}{\alpha+1}-2\,\frac{\alpha\, m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,t)}{\alpha+1}+\frac{m\,\sigma(t-1)}{k\,(\alpha+1)}$$

$$-\frac{m\,\sigma(t-1)\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,(t-1))}{k\,(\alpha+1)}-\frac{\sigma(t-1)\,\alpha\, m}{k\,(\alpha+1)^2}+\frac{1}{2}\,\frac{\sigma(t-1)\,t^2}{\alpha+1}-\frac{\sigma(t-1)\,t}{\alpha+1}$$

$$\left.+\frac{1}{2}\,\frac{\sigma(t-1)}{\alpha+1}+\frac{\sigma(t-1)\,\alpha\, m\cos(\sqrt{\frac{k\,(\alpha+1)}{\alpha\, m}}\,(t-1))}{k\,(\alpha+1)^2}\right)\Big/m\Bigg\}$$

Für die Konstanten setzen wir wieder die obigen Werte ein.

```
> sol2 := collect(eval(%, {alpha=1/10,m=1,k=1}), sigma(t-1));
```

$$sol2:=\{x_2(t)=(\frac{45}{121}+\frac{5}{11}\,t^2-\frac{10}{11}\,t+\frac{10}{121}\cos(\sqrt{11}\,(t-1)))\,\sigma(t-1)+\frac{2}{11}-\frac{2}{11}\cos(\sqrt{11}\,t),$$
$$x_1(t)=(\frac{155}{121}-\frac{100}{121}\cos(\sqrt{11}\,(t-1))+\frac{5}{11}\,t^2-\frac{10}{11}\,t)\,\sigma(t-1)+\frac{20}{11}\cos(\sqrt{11}\,t)+\frac{2}{11}\}$$

Diese speziellen Lösungen definieren wir nun als Maple-Funktionen $z_1$und $z_2$

```
> z[1] := unapply(eval(x[1](t), sol2), t);
```

$$z_1:=t\to(\frac{155}{121}-\frac{100}{121}\cos(\sqrt{11}\,(t-1))+\frac{5}{11}\,t^2-\frac{10}{11}\,t)\,\sigma(t-1)+\frac{20}{11}\cos(\sqrt{11}\,t)+\frac{2}{11}$$

```
> z[2] := unapply(eval(x[2](t), sol2), t);
```

$$z_2:=t\to(\frac{45}{121}+\frac{5}{11}\,t^2-\frac{10}{11}\,t+\frac{10}{121}\cos(\sqrt{11}\,(t-1)))\,\sigma(t-1)+\frac{2}{11}-\frac{2}{11}\cos(\sqrt{11}\,t)$$

und vergleichen sie dann mit den dsolve Lösungsfunktionen $y_1$ und $y_2$.

```
> simplify(z[1](t)-y[1](t));
```

$$0$$

```
> simplify(z[2](t)-y[2](t));
```

$$0$$

## 6.3 Näherungslösung durch Reihenentwicklung

```
> restart:
```

Eine weitere mögliche Option von dsolve lautet series. Maple führt dann für die betreffenden Funktionen eine Reihenentwicklung durch und liefert die Lösung der Differentialgleichung als Reihe. Diese Vorgehensweise hat den Vorteil, daß unter Umständen auch dann eine symbolische Lösung berechnet werden kann, wenn dsolve eine exakte Lösung nicht mehr findet. Der Nachteil der series Option besteht darin, daß die resultierende Lösung nur eine Näherungslösung ist, die die exakte Lösung nur in einer (zumeist kleinen) Umgebung des Entwicklungspunktes gut approximiert. Beispiel:

```
> de1 := diff(y(t),t$2) + diff(y(t),t) + y(t) = t + sin(t);
```

$$de1 := (\frac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + (\frac{\partial}{\partial t}\,\mathrm{y}(t)) + \mathrm{y}(t) = t + \sin(t)$$

```
> ini1 := y(0)= 0, D(y)(0) = 0;
```

$$ini1 := \mathrm{y}(0) = 0,\ \mathrm{D}(y)(0) = 0$$

Die Anzahl der Terme der Reihenentwicklung wird durch die globale Variable **Order** bestimmt, die mit dem Wert 6 voreingestellt ist.

```
> Order := 10:
> dsolve({de1,ini1}, y(t), series);
```

$$\mathrm{y}(t) = \frac{1}{3}\,t^3 - \frac{1}{12}\,t^4 - \frac{1}{120}\,t^5 + \frac{1}{240}\,t^6 - \frac{1}{5040}\,t^7 - \frac{1}{20160}\,t^8 + \frac{1}{181440}\,t^9 + \mathrm{O}(t^{10})$$

```
> f[1] := unapply(convert(rhs(%), polynom), t);
```

$$f_1 := t \to \frac{1}{3}\,t^3 - \frac{1}{12}\,t^4 - \frac{1}{120}\,t^5 + \frac{1}{240}\,t^6 - \frac{1}{5040}\,t^7 - \frac{1}{20160}\,t^8 + \frac{1}{181440}\,t^9$$

Diese Differentialgleichung kann Maple auch exakt lösen.

```
> dsolve({de1,ini1}, y(t)):
> combine(%, trig);
```

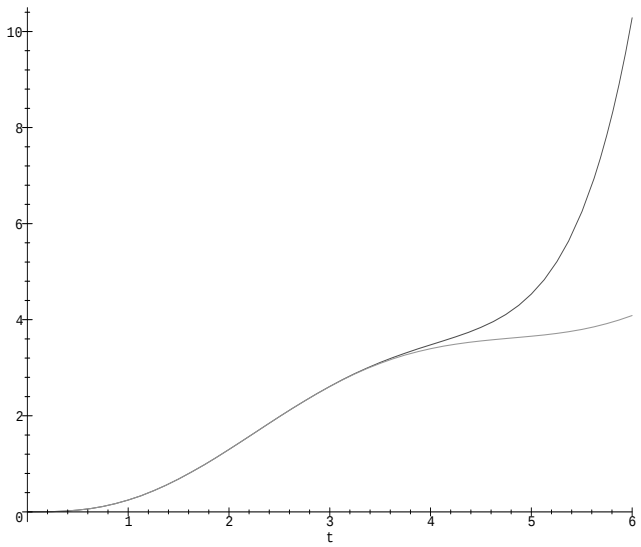
$$\mathrm{y}(t) = -\cos(t) + 2\,e^{(-1/2\,t)}\cos(\frac{1}{2}\,\sqrt{3}\,t) + t - 1$$

```
> f[2] := unapply(rhs(%), t);
```

$$f_2 := t \to -\cos(t) + 2\,e^{(-1/2\,t)}\cos(\frac{1}{2}\,\sqrt{3}\,t) + t - 1$$

Die folgende Abbildung zeigt die Kurven der tatsächlichen Lösung und der Näherungslösung.

```
> plot({f[1](t),f[2](t)}, t=0..6);
```




Wenn die Differentialgleichung symbolisch nicht gelöst werden kann, gibt Maple manchmal eine Darstellung in Form von **DESol** Termen zurück. Die Datenstruktur DESol ist mit RootOf vergleichbar und dient als Platzhalter für die Lösung der Differentialgleichung. Beispiel:

```
> de2 := diff(y(t), t$3) + (2*t+2)*diff(y(t), t$2) +
(4*t+4-1/t)*diff(y(t), t) + y(t);
```

$$de2 := (\frac{\partial^3}{\partial t^3}\,\mathrm{y}(t)) + (2\,t+2)\,(\frac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + (4\,t+4-\frac{1}{t})\,(\frac{\partial}{\partial t}\,\mathrm{y}(t)) + \mathrm{y}(t)$$

```
> dsolve(de2, y(t));
```

Beispiel:

```
> de2 := diff(y(t), t$3) + (2*t+2)*diff(y(t), t$2) +
(4*t+4-1/t)*diff(y(t), t) + y(t);
```

$$de2 := (\frac{\partial^3}{\partial t^3}\,\mathrm{y}(t)) + (2\,t+2)\,(\frac{\partial^2}{\partial t^2}\,\mathrm{y}(t)) + (4\,t+4-\frac{1}{t})\,(\frac{\partial}{\partial t}\,\mathrm{y}(t)) + \mathrm{y}(t)$$

```
> dsolve(de2, y(t));
```

$$\mathrm{y}(t) = \mathrm{DESol}(\{t\,(\frac{\partial^3}{\partial t^3}\,\_\mathrm{Y}(t)) + (2\,t^2+2\,t)\,(\frac{\partial^2}{\partial t^2}\,\_\mathrm{Y}(t)) + (4\,t^2+4\,t-1)\,(\frac{\partial}{\partial t}\,\_\mathrm{Y}(t)) + t\,\_\mathrm{Y}(t)\},$$
$$\{\_\mathrm{Y}(t)\})$$

Diese Lösung kann man jetzt mit dem Kommando **series** in eine Reihe entwickeln:

```
> y(t) = series(rhs(%), t=0, 4);
```

$$\mathrm{y}(t) = -4\,\_C3 + (12\,\_C3 + 12\,\_C2)\,t + (\_C3\,(-6+6\ln(t)) + \_C1 + \_C2\,(6\ln(t)+5))\,t^2$$
$$+(\_C3\,(-\frac{20}{3} - 2\ln(t)) - \frac{1}{3}\,\_C1 + \_C2\,(-2\ln(t)-11))\,t^3 + \mathrm{O}(t^4)$$

## 6.4 Numerische Lösung

```
> restart:
```

Wenn dsolve keine symbolische Lösung der Differentialgleichung findet, besteht noch die Hoffnung, daß mit der Option numeric (**dsolve**,**numeric**) eine numerische Behandlung der Gleichung möglich ist. Voraussetzung für den Einsatz der Option numeric ist allerdings, daß die Differentialgleichung keine symbolischen Konstanten enthält und ausreichend Rand- und Nebenbedingungen formuliert werden können. dsolve bietet mit der Option numeric verschiedene Ausgabemöglichkeiten. Standardmäßig (output=procedurelist) gibt dsolve eine numerische Lösungsprozedur zurück. Beispiel:

```
> de1 := diff(x(t),t) = y(t), diff(y(t),t) = x(t)+y(t);
```

$$de1 := \frac{\partial}{\partial t}\,\mathrm{x}(t) = \mathrm{y}(t),\ \frac{\partial}{\partial t}\,\mathrm{y}(t) = \mathrm{x}(t) + \mathrm{y}(t)$$

```
> ini1 := x(0) = 2, y(0) = 1;
```

$$ini1 := \mathrm{x}(0) = 2,\ \mathrm{y}(0) = 1$$

```
> sol1 := dsolve({de1,ini1}, {x(t),y(t)}, type=numeric);
```

$$sol1 := \mathbf{proc}(rkf45\_x)\ \ldots\ \mathbf{end}$$

Für einzelne Zeitpunkte berechnet diese Lösungsprozedur eine Ergebnisliste.

```
> sol1(0);
```

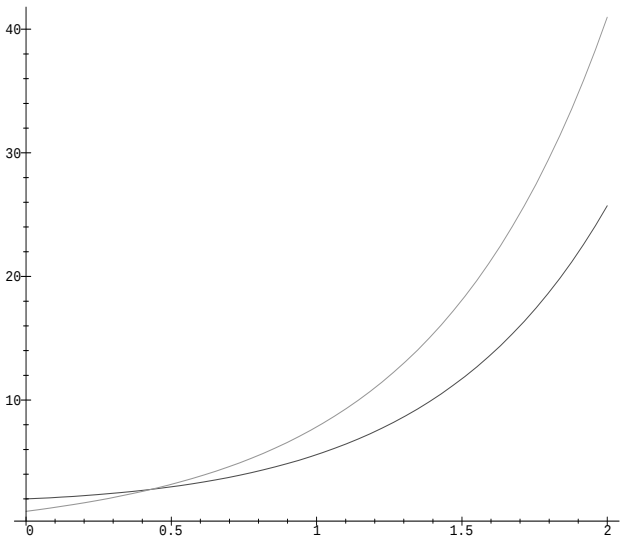
$$[t = 0,\ \mathrm{y}(t) = 1.,\ \mathrm{x}(t) = 2.]$$

```
> sol1(1);
```

$$[t = 1,\ \mathrm{y}(t) = 7.826891137110795,\ \mathrm{x}(t) = 5.582168689244845]$$

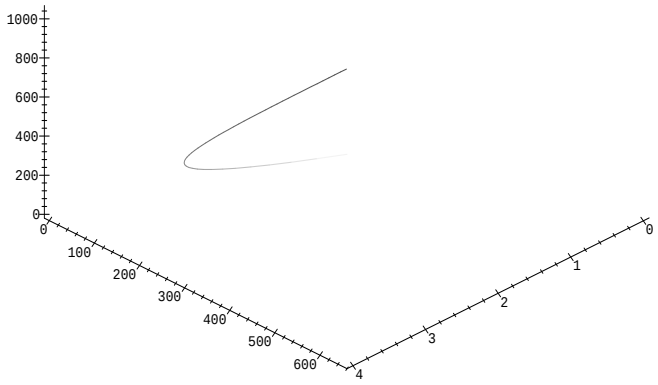
Mit der Funktion odeplot aus dem Graphikpaket plots lassen sich die numerischen Lösungen direkt veranschaulichen.

```
> with(plots):
> odeplot(sol1, [[t,x(t)], [t,y(t)]], 0..2);
```




Die gleiche Lösung als Kurve im Raum:

```
> odeplot(sol1, [t,x(t),y(t)], 0..4, axes=frame);
```




Mit output=listprocedure wird eine Liste von Gleichungen erzeugt. Unabhängige und abhängige Variablen stehen auf der linken Seite, Prozeduren stehen auf der rechten Seite.

```
> sol2 := dsolve({de1,ini1}, {x(t),y(t)}, type=numeric,
output=listprocedure);
```

$$sol2 := [t = (\mathbf{proc}(t) \ldots \mathbf{end}), \mathrm{x}(t) = (\mathbf{proc}(t) \ldots \mathbf{end}), \mathrm{y}(t) = (\mathbf{proc}(t) \ldots \mathbf{end})]$$

Durch Substitution lassen sich die Lösungsprozeduren aus der Liste extrahieren und dann plotten.

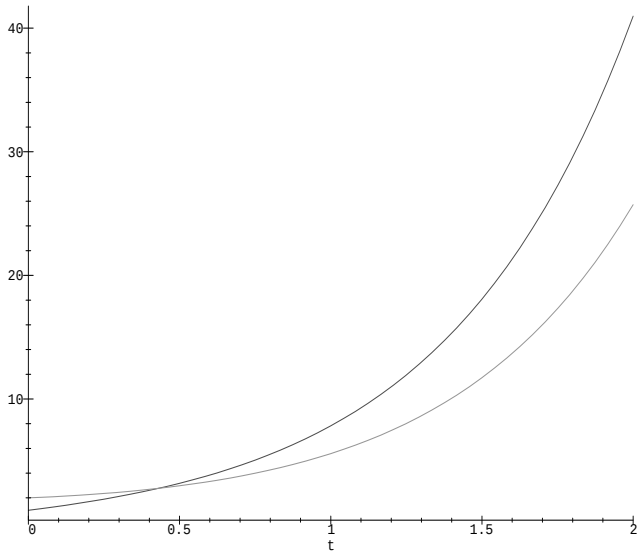
```
> fx := eval(x(t), sol2);
```

$$fx := \mathbf{proc}(t) \ldots \mathbf{end}$$

```
> fy := eval(y(t), sol2);
```

$$fy := \mathbf{proc}(t) \ldots \mathbf{end}$$

```
> plot({'fx(t)', 'fy(t)'}, t=0..2);
```



Hier wird die symbolische Auswertung der numerischen Prozeduren fx und fy durch die Aposthrophierung verhindert. Mit der Option value=array([...]) werden die Lösungsfunktionen an vorgegebenen Punkten berechnet und die numerischen Werte in Form einer Tabelle (Matrix) ausgegeben.

```
> F := dsolve({de1,ini1}, {x(t),y(t)}, type=numeric,
> value=array([0,.6,1.1,1.5,2.3,2.5]));
```

$$F := \left[ \begin{array}{c} [t,\, \mathrm{y}(t),\, \mathrm{x}(t)] \\ \left[ \begin{array}{ccc} 0 & 1. & 2. \\ .6 & 3.845239673 & 3.330277377 \\ 1.1 & 9.279991644 & 6.435591574 \\ 1.5 & 18.08036908 & 11.72115280 \\ 2.3 & 66.71679118 & 41.56679172 \\ 2.5 & 92.28386863 & 57.32933214 \end{array} \right] \end{array} \right]$$

Neben diesen numerischen Lösungsmöglichkeiten hat Maple noch ein eigenes Paket **DEtools** für die graphische Analyse von Differentialgleichungen.

# Kapitel 7

# Lineare Algebra

## 7.1 Vektoroperationen

```
> restart:
```

Die Befehle zur linearen Algebra befinden sich im Paket **linalg** und müssen zunächst aktiviert werden:

```
> with(linalg);
```

```
Warning, new definition for norm

Warning, new definition for trace
```

$$[BlockDiagonal, GramSchmidt, JordanBlock, LUdecomp, QRdecomp, Wronskian, addcol, addrow, adj, adjoint, angle, augment, backsub, band, basis, bezout, blockmatrix, charmat, charpoly, cholesky, col, coldim, colspace, colspan, companion, concat, cond, copyinto, crossprod, curl, definite, delcols, delrows, det, diag, diverge, dotprod, eigenvals, eigenvalues, eigenvectors, eigenvects, entermatrix, equal, exponential, extend, ffgausselim, fibonacci, forwardsub, frobenius, gausselim, gaussjord, geneqns, genmatrix, grad, hadamard, hermite, hessian, hilbert, htranspose, ihermite, indexfunc, innerprod, intbasis, inverse, ismith, issimilar, iszero, jacobian, jordan, kernel, laplacian, leastsqrs, linsolve, matadd, matrix, minor, minpoly, mulcol, mulrow, multiply, norm, normalize, nullspace, orthog, permanent, pivot, potential, randmatrix, randvector, rank, ratform, row, rowdim, rowspace, rowspan, rref, scalarmul, singularvals, smith, stackmatrix, submatrix, subvector, sumbasis, swapcol, swaprow, sylvester, toeplitz, trace, transpose, vandermonde, vecpotent, vectdim, vector, wronskian]$$

In Maple ist ein Vektor ein eindimensionales Feld (**array**), dessen Index mit 1 beginnt :

```
> V := vector(5);
```

$$V := \mathrm{array}(1..5, [])$$

```
> print(V);
```

$$[V_1, V_2, V_3, V_4, V_5]$$

Vektoren können direkt als eindimensionales Feld oder mit dem **vector** Kommando eingegeben werden:

```
> V1 := array([1/2,5/2,9]);
```

$$V1 := \left[\frac{1}{2}, \frac{5}{2}, 9\right]$$

```
> V2 := vector([a,b,c]);
```

$$V2 := [a, b, c]$$

Erfüllen die Elemente eines Vektors eine vorgegebene Gesetzmäßigkeit, so kann man den Vektor mit Hilfe einer anonymen Prozedur erzeugen:

```
> V3 := vector(3, n -> sin(n*Pi/2));
```

$$V3 := [1, 0, -1]$$

Die Länge des Vektors wird hier durch den ersten Parameter des vector Kommandos festgelegt. Vektoren bis zur Länge 4 kann man auch über die neu eingeführte Matrix-Palette bequem eingeben.

Einzelne Vektor-Komponenten können durch die Angabe des Index in eckigen Klammern angesprochen werden:

```
> V[2]; V3[3];
```

$$V_2$$
$$-1$$

Zur Auswertung von Vektoren (allgemeiner des Datentyps **array** ) in Maple ist zu bemerken, daß sie nach speziellen Regeln (wie bei Prozeduren) ausgewertet werden. Um die Datenstruktur des Vektors $V3$ darzustellen, genügt nach der Definition von $V3$ nicht die Eingabe

```
> V3;
```

$$V3$$

denn dann wird nur der Name $V3$ zurückgegeben. Die Ausgabe von $V3$ wird durch die Kommandos

```
> eval(V3); op(V3);
```

$$[1,\ 0,\ -1]$$
$$[1,\ 0,\ -1]$$

oder

```
> print(V3);
```

$$[1,\ 0,\ -1]$$

veranlaßt.

Die Länge bzw. der Betrag eines Vektors wird durch das **norm** Kommando berechnet:

```
> norm(V2,2); norm(V3,2);
```

$$\sqrt{|a|^2 + |b|^2 + |c|^2}$$
$$\sqrt{2}$$

Die Ausführung der Addition zweier Vektoren und die Multiplikation eines Vektors mit einem Skalar erfolgt mit dem **evalm** Befehl:

```
> evalm(V + 1);
> evalm(2*V2);
> evalm(V1 + V2);
```

$$[V_1 + 1,\ V_2 + 1,\ V_3 + 1,\ V_4 + 1,\ V_5 + 1]$$
$$[2\,a,\ 2\,b,\ 2\,c]$$
$$\left[\frac{1}{2} + a,\ \frac{5}{2} + b,\ 9 + c\right]$$

Das Skalarprodukt wird durch das **dotprod** Kommando (Punktprodukt) realisiert:

```
> dotprod(V1,V2);
```

$$\frac{1}{2}\,\overline{a} + \frac{5}{2}\,\overline{b} + 9\,\overline{c}$$

Wenn der zweite Vektor komplexe Werte enthält, bildet dotprod vor der Multiplikation automatisch die konjugiert komplexen Werte. Durch die Option orthogonal kann die Bildung konjugiert komplexer Werte vermieden werden:

```
> dotprod(V1,V2, orthogonal);
```

$$\frac{1}{2}\,a + \frac{5}{2}\,b + 9\,c$$

Der Winkel zwischen zwei Vektoren läßt sich mit der **angle** Funktion berechnen:

```
> angle(V1,V3);
```

$$\pi - \arccos(\frac{17}{350}\sqrt{175})$$

```
> evalf(%);
```

$$2.268604324$$

Das Ergebnis wird im Bogenmaß angegeben; mit **convert**,**degrees** erhält man die Angabe in Grad:

```
> evalf(convert(%,degrees));
```

$$129.9814531\ degrees$$

Für das Kreuzprodukt (Vektorprodukt) steht der **crossprod** Befehl zur Verfügung:

```
> crossprod(V1,V2);
```

$$\left[\frac{5}{2}\,c - 9\,b,\ 9\,a - \frac{1}{2}\,c,\ \frac{1}{2}\,b - \frac{5}{2}\,a\right]$$

Beim Ergebnis des Kreuzproduktes ist die horizontale Schreibweise des Vektors unüblich und unübersichtlich. Durch Konvertieren in eine 3x1-Matrix erhält man den entsprechenden Spaltenvektor:

```
> convert(%, matrix);
```

$$\begin{bmatrix} \frac{5}{2}\,c - 9\,b \\ 9\,a - \frac{1}{2}\,c \\ \frac{1}{2}\,b - \frac{5}{2}\,a \end{bmatrix}$$

Mit Hilfe des Kreuzproduktes berechnet sich der Flächeninhalt des von zwei Vektoren aufgespannten Parallelogramms wie folgt:

```
> cp := crossprod(V1, V3);
> Flaeche := evalf(norm(cp,2));
```

$$cp := \left[\frac{-5}{2},\ \frac{19}{2},\ \frac{-5}{2}\right]$$

$$Flaeche := 10.13656747$$

Mit **normalize** kann man die Länge eines Vektors (gemessen in der 2-Norm) auf 1 normieren.

```
> normalize(V1);
```

$$\left[\frac{1}{70}\sqrt{7}\sqrt{2},\ \frac{1}{14}\sqrt{7}\sqrt{2},\ \frac{9}{35}\sqrt{7}\sqrt{2}\right]$$

```
> norm(%,2);
```

$$1$$

Das Spatprodukt läßt sich als Kombination von Skalarprodukt und Kreuzprodukt darstellen. Für das Volumen eines Spates fogt dann:

```
> A := vector(3): B:=vector(3): C:=vector(3):
> Vol := abs(dotprod(A, crossprod(B,C),orthogonal));
```

$$Vol := |A_1\,(B_2\,C_3 - B_3\,C_2) + A_2\,(B_3\,C_1 - B_1\,C_3) + A_3\,(B_1\,C_2 - B_2\,C_1)|$$

## 7.2 Matrizenrechnung

```
> restart:
> with(linalg):

Warning, new definition for norm

Warning, new definition for trace
```

Eine Matrix wird in Maple durch ein zweidimensionales Feld repräsentiert, dessen Zeilen- und Spaltenindizes mit 1 beginnen. Das Standardkommando zur Definition einer Matrix lautet **matrix**. Dem Kommando kann neben den gewünschten Dimensionen der Matrix eine Liste mit den Matrixelementen übergeben werden.

```
> A := matrix(3,2,[a,b,c,d,e,f]);
```

$$A := \begin{bmatrix} a & b \\ c & d \\ e & f \end{bmatrix}$$

Wenn die Matrixelemente in einer verschachtelten Liste übergeben werden, ermittelt Maple die Dimension selbstständig:

```
> matrix([[1,1],[2,2],[3,3]]);
```

$$\begin{bmatrix} 1 & 1 \\ 2 & 2 \\ 3 & 3 \end{bmatrix}$$

Die Listen werden hierbei als Zeilen der Matrix interpretiert. Die Initialisierung der Matrixelemente kann auch über eine zweiparametrige anonyme Funktion erfolgen. Der erste Parameter dieser Funktion enthält den Zeilenindex, der zweite Parameter den Spaltenindex.

```
> matrix(3,3,(n,m) -> n^m);
```

$$\begin{bmatrix} 1 & 1 & 1 \\ 2 & 4 & 8 \\ 3 & 9 & 27 \end{bmatrix}$$

Für die Erzeugung von Spezialmatrizen stehen zahlreiche Kommandos in linalg (**diag**, **band**, **toeplitz**, **hilbert**, **randmatrix**, ...) und spezielle Indexfunktionen des **array** Kommandos zur Verfügung, z. B.

```
> Null_Matrix := matrix(3,3,0);
```

$$Null\_Matrix := \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

```
> array(1..3,1..3,identity);
```

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

```
> Diagonal_Matrix := diag(1,2,3);
```

$$Diagonal\_Matrix := \begin{bmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 3 \end{bmatrix}$$

```
> Band_Matrix := band([-1,1,1],4);
```

$$Band\_Matrix := \begin{bmatrix} 1 & 1 & 0 & 0 \\ -1 & 1 & 1 & 0 \\ 0 & -1 & 1 & 1 \\ 0 & 0 & -1 & 1 \end{bmatrix}$$

```
> Toeplitz_Matrix := toeplitz([1,2,3,4]);
```

$$Toeplitz\_Matrix := \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 2 & 3 \\ 3 & 2 & 1 & 2 \\ 4 & 3 & 2 & 1 \end{bmatrix}$$

```
> Hilbert_Matrix := hilbert(3);
```

$$Hilbert\_Matrix := \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$

```
> Zufallszahlen_Matrix := randmatrix(3,3, entries=rand(0..10));
```

$$Zufallszahlen\_Matrix := \begin{bmatrix} 9 & 4 & 1 \\ 4 & 4 & 1 \\ 3 & 8 & 9 \end{bmatrix}$$

Durch die Option entries=rand(0..10) wird das Intervall für die Zufallszahlen festgelegt. Der Zugriff auf ein einzelnes Matrixelement erfolgt durch die Angabe der Indizes (zuerst Zeile, dann Spalte):

```
> A[1,1];
```

$$a$$

Für den Zugriff auf Teilmatrizen gibt es das **submatrix** Kommando:

```
> submatrix(A,2..3,1..2);
```

$$\begin{bmatrix} c & d \\ e & f \end{bmatrix}$$

Das Lesen ganzer Zeilen und Spalten erfolgt mit den Befehlen **row** und **col** :

```
> col(A,1);
```

$$[a, c, e]$$

```
> row(A,2..3);
```

$$[c, d], [e, f]$$

Addition und Subtraktion von Matrizen und die Multiplikation einer Matrix mit einem Skalar werden mit +, -, * gekennzeichnet; die Matrixmultiplikation wird mit dem Operator &* dargestellt, da "*" für die kommutative Multiplikation reserviert ist. Durch **evalm** werden die Rechenoperationen ausgeführt.

```
> B := matrix ([[r,s,t],[u,v,w]]);
```

$$B := \begin{bmatrix} r & s & t \\ u & v & w \end{bmatrix}$$

```
> evalm(A &* B) <> evalm(B &* A);
```

$$\begin{bmatrix} ar+bu & as+bv & at+bw \\ cr+du & cs+dv & ct+dw \\ er+fu & es+fv & et+fw \end{bmatrix} \neq \begin{bmatrix} ar+cs+et & rb+sd+tf \\ ua+vc+we & bu+dv+fw \end{bmatrix}$$

**transpose** transponiert eine Matrix, d.h. die Elemente werden entlang der Hauptdiagonalen vertauscht:

```
> transpose(B);
```

$$\begin{bmatrix} r & u \\ s & v \\ t & w \end{bmatrix}$$

```
> evalm(2*A - 3*transpose(B));
```

$$\begin{bmatrix} 2a-3r & 2b-3u \\ 2c-3s & 2d-3v \\ 2e-3t & 2f-3w \end{bmatrix}$$

Die Addition eines Skalars führt zur Addition der Einheitsmatrix multipliziert mit dem Skalar:

```
> evalm(B + 2);
```

$$\begin{bmatrix} r+2 & s & t \\ u & v+2 & w \end{bmatrix}$$

Die Potenz einer Matrix entspricht der wiederholten Matrixmultiplikation:

```
> M := matrix([[1,2],[3,4]]);
```

$$M := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> evalm(M^3) = evalm(M &* M &* M);
```

$$\begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix} = \begin{bmatrix} 37 & 54 \\ 81 & 118 \end{bmatrix}$$

Die Determinante einer Matrix wird mit **det** und die Inverse einer Matrix mit **inverse** berechnet.

```
> det(M);
```

$$-2$$

```
> Minv := inverse(M);
```

$$Minv := \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix}$$

```
> eval(M)*eval(Minv) = evalm(M &* Minv);
```

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} \begin{bmatrix} -2 & 1 \\ \frac{3}{2} & \frac{-1}{2} \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

## 7.3 Lineare Gleichungssysteme

```
> restart:
> with(linalg):

Warning, new definition for norm

Warning, new definition for trace
```

Die Konditionszahl (**cond**) einer Matrix

```
> A := matrix(3,3, [1,0,3,-4,2,0,0,3,-2]);
```

$$A := \begin{bmatrix} 1 & 0 & 3 \\ -4 & 2 & 0 \\ 0 & 3 & -2 \end{bmatrix}$$

wird über die Matrixnorm nach der Formel

```
> 'cond(A)' = norm(A)*norm(inverse(A));
```

$$\mathrm{cond}(A) = \frac{33}{10}$$

berechnet. Sie spielt eine große Rolle bei linearen Gleichungssystemen $A\,x = b$ mit fehlerbehaftetem $b$. Eine Änderung $\Delta\,b$ hat eine Änderung der Lösung $\Delta\,x$ zur Folge. Es läßt sich zeigen, daß der relative Fehler von x kleiner oder gleich dem relativen Fehler von b multipliziert mit der Konditionszahl *cond(A)* ist.

$$\frac{|\Delta\,x|}{|x|} \leq \frac{|\Delta\,b|}{|b|}\ \mathit{cond(A)}$$

Ein Gleichungssystem ist daher schlecht konditioniert, wenn die Konditionszahl viel größer als 1 ist. Beispiel: Betrachten wir die Polynominterpolation durch vorgegebene Punkte $(x_1,\, y_1)$, $(x_2,\, y_2)$, ..., $(x_{n+1},\, y_{n+1})$ mit einem Polynom $P_n(x) = a_0$ + ...+ $a_n\,x^n$ *n*-ten Grades, so erhalten wir die Bestimmungsgleichungen $P_n(x_i) = y_i$ für die gesuchten Polynomkoeffizienten $a_0,\, a_1$,..., $a_n$ in Matrixform $A\,a = y$ aus:

```
> n := 5:
> x := vector(5):
> A := vandermonde(convert(x,list));
```

$$A := \begin{bmatrix} 1 & x_1 & {x_1}^2 & {x_1}^3 & {x_1}^4 \\ 1 & x_2 & {x_2}^2 & {x_2}^3 & {x_2}^4 \\ 1 & x_3 & {x_3}^2 & {x_3}^3 & {x_3}^4 \\ 1 & x_4 & {x_4}^2 & {x_4}^3 & {x_4}^4 \\ 1 & x_5 & {x_5}^2 & {x_5}^3 & {x_5}^4 \end{bmatrix}$$

Die Koeffizientenmatrix hat die Form einer Vandermonde-Matrix (**vandermonde**).

```
> x := vector(n, i->1+i*0.1);
```

$$x := [1.1,\ 1.2,\ 1.3,\ 1.4,\ 1.5]$$

Für die Auswertung der Matrixelemente von $A$ muß die Funktion **map**, die eine Prozedur/Funktion auf jeden Operanden eines Ausdrucks ausführt, angewandt werden.

```
> A := map(eval, A);
```

$$A := \begin{bmatrix} 1 & 1.1 & 1.21 & 1.331 & 1.4641 \\ 1 & 1.2 & 1.44 & 1.728 & 2.0736 \\ 1 & 1.3 & 1.69 & 2.197 & 2.8561 \\ 1 & 1.4 & 1.96 & 2.744 & 3.8416 \\ 1 & 1.5 & 2.25 & 3.375 & 5.0625 \end{bmatrix}$$

$A$ ist eine schlecht konditionierte Matrix, d.h. kleine Meßunsicherheiten im Datenvektor $y$ rufen große Änderungen $\Delta\,a$ in den Polynomkoeffizienten hervor.

```
> cond(A,frobenius);
```

$$539487.2910$$

*cond(A, frobenius)* ist die Euklidische Matrixnorm (Wurzel aus der Betragsquadratsumme aller Matrixelemente).

```
> y1 := vector(n, i ->sin(1+i*0.1));
```

$$y1 := [.8912073601,\ .9320390860,\ .9635581854,\ .9854497300,\ .9974949866]$$

Wir lösen nun das lineare Gleichungssystem $A\,x = y1$ mit dem Kommando **linsolve** :

```
> a1 := linsolve(A, y1);
```

$$a1 := [.0137849717,\ .9424184720,\ .0980825708,\ -.2528928521,\ .0400809675]$$

Der Vektor $y1$ wird nun ein wenig verändert:

```
> y2:=vector(n,[y1[1]*1.01,y1[2]*0.99, y1[3]*0.995,y1[4]*1.003,
y1[5]*1.002]);
```

$$y2 := [.9001194337,\ .9227186951,\ .9587403945,\ .9884060792,\ .9994899766]$$

```
> a2 := linsolve(A, y2);
```

$$a2 := [14.75426325,\ -39.91392473,\ 41.96212164,\ -19.03113123,\ 3.146944799]$$

```
> norm(y1-y2,frobenius)/norm(y1,frobenius);
```

$$.006661239060$$

```
> % * cond(A,frobenius);
```

$$3593.653815$$

```
> norm(a1-a2,frobenius)/norm(a1,frobenius);
```

$$64.44280458$$

Die Ungleichung

$$\frac{|\Delta\, a|}{|a|} \le \frac{|\Delta\, y|}{|y|}\ cond(A)$$

ist zwar immer noch erfüllt, aber die Schwankungen $\Delta\, a$ der Poynomkoeffizienten sind inakzeptabel groß.

Nun wollen wir die Gaußelimination zur Lösung linearer Gleichungssysteme verwenden. Dabei formt man die Koeffizientenmatrix durch Zeilenoperationen in eine obere Dreiecksmatrix um und wendet auf *b* dieselben Zeilenoperationen an. Mittels Rückwärtseinsetzen erhält man dann die gesuchte Lösung. Beispiel:

```
> A := matrix([[3,-13,9,3],[-6,4,1,-18],[6,-2,2,4], [12,-8,6,10]]);
```

$$A := \begin{bmatrix} 3 & -13 & 9 & 3 \\ -6 & 4 & 1 & -18 \\ 6 & -2 & 2 & 4 \\ 12 & -8 & 6 & 10 \end{bmatrix}$$

```
> b := vector([-19,-34,16,26]);
```

$$b := [-19,\ -34,\ 16,\ 26]$$

Die Matrix *A* und der Vektor *b* werden nun mit **augment** horizontal zur erweiterten Matrix Matrix *B* zusammengefügt.

```
> B := augment(A,b);
```

$$B := \begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ -6 & 4 & 1 & -18 & -34 \\ 6 & -2 & 2 & 4 & 16 \\ 12 & -8 & 6 & 10 & 26 \end{bmatrix}$$

```
> L := gausselim(B, 'rangA', 'detA');
```

$$L := \begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ 0 & -22 & 19 & -12 & -72 \\ 0 & 0 & 8 & -26 & -42 \\ 0 & 0 & 0 & \frac{3}{11} & \frac{3}{11} \end{bmatrix}$$

```
> ffgausselim(B, 'rangA', 'detA');
```

$$\begin{bmatrix} 3 & -13 & 9 & 3 & -19 \\ 0 & -66 & 57 & -36 & -216 \\ 0 & 0 & -312 & 996 & 1620 \\ 0 & 0 & 0 & 144 & 144 \end{bmatrix}$$

Wenn statt der Funktion **gausselim ffgausselim** (ff steht für fraction free) verwendet wird, ist die berechnete Dreiecksmatrix bruchfrei. Die Funktion **backsub** führt das Rückwärtseinsetzen durch und liefert den gesuchten Lösungsvektor:

```
> x := backsub(L);
```

$$x := [3,\ 1,\ -2,\ 1]$$

Die optionalen Parameter beim Aufruf dienen zur gleichzeitigen Berechnung von Rang und Determinante der Matrix *A* .

```
> rangA;
```

$$4$$

```
> detA;
```

$$144$$

Eine der Gaußelimination verwandte Methode ist das Gauß-Jordan-Verfahren (**gaussjord** ist identisch mit der reduced row echelon form **rref**). Bei dieser Methode wird die Matrix auf Diagonalform gebracht; damit erspart man sich das Rückwärtseinsetzen, der Lösungsvektor steht in der letzten Spalte der Matrix:

```
> gaussjord(B);
```

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 3 \\ 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & -2 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix}$$

Wenn die Determinante verschwindet, ist das Gleichungssystem entweder nicht lösbar, oder die Lösung enthält noch freie Parameter.

```
> A := matrix(3,3,[1,2,3,4,5,6,7,8,9]);
```

$$A := \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
> det(A);
```

$$0$$

```
> b := vector([1,2,3]);
```

$$b := [1,\ 2,\ 3]$$

```
> gausselim(augment(A,b));
```

$$\begin{bmatrix} 1 & 2 & 3 & 1 \\ 0 & -3 & -6 & -2 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

```
> backsub(%);
```

$$\left[-\frac{1}{3} + \_t_1,\ \frac{2}{3} - 2\,\_t_1,\ \_t_1\right]$$

Die Lösung besitzt einen freien Parameter $\_t_1$.

## 7.4 Diagonalisierung

```
> restart:
> with(linalg):

Warning, new definition for norm

Warning, new definition for trace
```

Funktionen zur Lösung des Eigenwertproblems $A\,x = \lambda\,x$ .

```
> A:=matrix(2,2,[1,2,3,4]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

Konstruktion der charakteristischen Matrix $\lambda\,I - A$ (**charmat**) :

```
> charmat(A,lambda);
```

$$\begin{bmatrix} \lambda - 1 & -2 \\ -3 & \lambda - 4 \end{bmatrix}$$

Das charakteristische Polynom ist somit

```
> det(%);
```

$$\lambda^2 - 5\,\lambda - 2$$

Direkt erhält man das charakteristische Polynom mit der Funktion **charpoly**

```
> P:=unapply(charpoly(A,lambda),lambda);
```

$$P := \lambda \to \lambda^2 - 5\,\lambda - 2$$

Die Eigenwerte werden dann durch Berechnung der Nullstellen des charakteristischen Polynoms bestimmt:

```
> solve(P(lambda)=0,lambda);
```

$$\frac{5}{2} + \frac{1}{2}\sqrt{33},\ \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

Direkt erhält man die Eigenwerte mit **eigenvals** :

```
> eigenvals(A);
```

$$\frac{5}{2} + \frac{1}{2}\sqrt{33},\ \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

Satz von Hamilton-Cayley:

```
> evalm(P(A));
```

$$\begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}$$

Für die numerische Berechnung der Eigenwerte gibt es die Funktion **Eigenvals** :

```
> evalf(Eigenvals(A,U));
```

$$[-.372281323,\ 5.372281323]$$

Die Matrix $U$ enthält die normierten Eigenvektoren von $A$ als Spaltenvektoren.

```
> evalm(U);
```

$$\begin{bmatrix} -.8245648401 & -.4222291504 \\ .5657674650 & -.9230523143 \end{bmatrix}$$

```
> det(U);
```

$$1.000000000$$

Durch die Matrixtransformation $U^{(-1)}\,A\,U$

```
> evalm(inverse(U) &* A &* U);
```

$$\begin{bmatrix} -.3722813231 & .3\,10^{-9} \\ 0 & 5.372281323 \end{bmatrix}$$

wird die Matrix $A$ in Diagonalform überführt. Ist $A$ symmetrisch, d.h. $A = A^T$, dann ist die Matrix $U$ orthogonal, d.h. $U^{(-1)} = U^T$.

Dies ist der Spektralsatz für symmetrische Matrizen: $\mathrm{D} = U^T A U$ .

Ein Nachteil der Anwendung von Eigenvals sind die Rundungsfehler!

Bestimmung der Eigenvektoren mit **eigenvects** :

```
> ev:=eigenvects(A,'radical');
```

$$ev := [\frac{5}{2} + \frac{1}{2}\sqrt{33},\ 1,\ \{\left[-\frac{1}{2} + \frac{1}{6}\sqrt{33},\ 1\right]\}],\ [\frac{5}{2} - \frac{1}{2}\sqrt{33},\ 1,\ \{\left[-\frac{1}{2} - \frac{1}{6}\sqrt{33},\ 1\right]\}]$$

Das Kommando eigenvects formuliert das Ergebnis als Folge, wobei jedes Element dieser Folge durch eine Liste bestehend aus Eigenwert, Vielfachheit und Eigenvektor formuliert wird.

```
> for i to 2 do
> lambda[i]:=ev[i][1];
> u[i]:=normalize(ev[i][3][1]);
> od;
```

$$\lambda_1 := \frac{5}{2} + \frac{1}{2}\sqrt{33}$$

$$u_1 := \left[6\,\frac{-\frac{1}{2} + \frac{1}{6}\sqrt{33}}{\sqrt{78 - 6\sqrt{33}}},\ 6\,\frac{1}{\sqrt{78 - 6\sqrt{33}}}\right]$$

$$\lambda_2 := \frac{5}{2} - \frac{1}{2}\sqrt{33}$$

$$u_2 := \left[6\,\frac{-\frac{1}{2} - \frac{1}{6}\sqrt{33}}{\sqrt{78 + 6\sqrt{33}}},\ 6\,\frac{1}{\sqrt{78 + 6\sqrt{33}}}\right]$$

Durch die Funktion **normalize** werden die Eigenvektoren auf die Länge 1 normiert.

Diagonalisieren der Matrix $A$: Wir bilden eine Matrix $U$, deren Spalten die normierten Eigenvektoren von $A$ sind:

```
> U:=transpose(matrix([u[1], u[2]]));
```

$$U := \begin{bmatrix} 6\,\frac{-\frac{1}{2} + \frac{1}{6}\sqrt{33}}{\sqrt{78 - 6\sqrt{33}}} & 6\,\frac{-\frac{1}{2} - \frac{1}{6}\sqrt{33}}{\sqrt{78 + 6\sqrt{33}}} \\ 6\,\frac{1}{\sqrt{78 - 6\sqrt{33}}} & 6\,\frac{1}{\sqrt{78 + 6\sqrt{33}}} \end{bmatrix}$$

Überführung in Diagonalform, die Diagonalelemente sind die Eigenwerte von $A$ :

```
> map(expand, evalm(inverse(U) &* A &* U));
```

$$\begin{bmatrix} \frac{5}{2} + \frac{1}{2}\sqrt{33} & 0 \\ 0 & \frac{5}{2} - \frac{1}{2}\sqrt{33} \end{bmatrix}$$

Lösung des verallgemeinerten Eigenwertproblems $A\,x = \lambda\,B\,x$ :

```
> A:=matrix(2,2,[1,2,3,4]);
```

$$A := \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

```
> B:=matrix(2,2,[1,1,2,4]);
```

$$B := \begin{bmatrix} 1 & 1 \\ 2 & 4 \end{bmatrix}$$

Bestimmung der Eigenwerte und Eigenvektoren mit Eigenvals:

```
> lambda:=evalf(Eigenvals(A,B,x));
```

$$\lambda := [1.280776406,\ -.7807764084]$$

```
> eval(x);
```

$$\begin{bmatrix} 1.000000000 & 1.000000000 \\ .3903882039 & -.6403882027 \end{bmatrix}$$

Bestimmung der normierten Eigenvektoren:

```
> x1:=normalize(col(x,1));
```

$$x1 := [.9315320879,\ .3636591387]$$

```
> x2:=normalize(col(x,2));
```

$$x2 := [.8421229403,\ -.5392855962]$$

Probe:

```
> evalm(A&*x1-lambda[1]*B&*x1);
```

$$[0,\ .1\,10^{-8}]$$

```
> evalm(A&*x2-lambda[2]*B&*x2);
```

$$[.22\,10^{-8},\ .21\,10^{-8}]$$

Bestimmung der exakten Eigenwerte mit eigenvals:

```
> lambda:=eigenvals(A,B);
```

$$\lambda := \frac{1}{4} + \frac{1}{4}\sqrt{17},\ \frac{1}{4} - \frac{1}{4}\sqrt{17}$$

```
> evalf(lambda);
```

$$1.280776407,\ -.7807764070$$

Leider ist es nicht möglich, mit einem Befehl wie

```
> eigenvects(A,B,'radical');

Error, (in factor) 2nd argument is not a valid algebraic extension, B
```

das verallgemeinerte Eigenwertproblem zu lösen. Da die Matrix $B$ allerdings regulär ist, können wir das verallgemeinerte Eigenwertproblem auf ein einfaches Eigenwertproblem zurückführen:

```
> det(B);
```

$$2$$

```
> ev:=eigenvects(evalm(inverse(B) &* A),'radical');
```

$$ev := [\frac{1}{4} + \frac{1}{4}\sqrt{17},\ 1,\ \{\left[\frac{1}{2} + \frac{1}{2}\sqrt{17},\ 1\right]\}],\ [\frac{1}{4} - \frac{1}{4}\sqrt{17},\ 1,\ \{\left[\frac{1}{2} - \frac{1}{2}\sqrt{17},\ 1\right]\}]$$

```
> lambda:='lambda':
> for i to 2 do
> lambda[i]:=ev[i][1];
> u[i]:=normalize(ev[i][3][1]);
> od;
```

$$\lambda_1 := \frac{1}{4} + \frac{1}{4}\sqrt{17}$$

$$u_1 := \left[2\,\frac{\frac{1}{2} + \frac{1}{2}\sqrt{17}}{\sqrt{22 + 2\sqrt{17}}},\ 2\,\frac{1}{\sqrt{22 + 2\sqrt{17}}}\right]$$

$$\lambda_2 := \frac{1}{4} - \frac{1}{4}\sqrt{17}$$

$$u_2 := \left[2\,\frac{\frac{1}{2} - \frac{1}{2}\sqrt{17}}{\sqrt{22 - 2\sqrt{17}}},\ 2\,\frac{1}{\sqrt{22 - 2\sqrt{17}}}\right]$$

```
> evalf(evalm(u[1]));
```

$$[.9315320884,\ .3636591382]$$

```
> evalf(evalm(u[2]));
```

$$[-.8421229398,\ .5392855962]$$

Wir erhalten bis auf die Richtung dieselben Eigenvektoren wie oben.

## 7.5 Matrixzerlegung

```
> restart:
> with(linalg):
```

```
Warning, new definition for norm
```

```
Warning, new definition for trace
```

In der numerischen Mathematik spielen Matrixzerlegungen wie die QR-Zerlegung, die Cholesky-Zerlegung und die Singulärwertzerlegung eine bedeutende Rolle.

1. QR-Zerlegung

Jede quadratische Matrix $A$ kann in eine orthonormale Matrix $Q$ mit $Q^T\,Q = I$ und eine obere Dreiecksmatrix $R$ der Form $A = Q\,R$ zerlegt werden.

```
> A:=matrix(3,3,[[1,1,1],[1,4,5],[2,3,1]]);
```

$$A := \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 5 \\ 2 & 3 & 1 \end{bmatrix}$$

Mit der Funktion **QRdecomp** läßt sich die obere Dreiecksmatrix berechnen:

```
> R := QRdecomp(A, Q='q');
```

$$R := \begin{bmatrix} \sqrt{6} & \frac{11}{6}\sqrt{6} & \frac{4}{3}\sqrt{6} \\ 0 & \frac{1}{6}\sqrt{210} & \frac{4}{15}\sqrt{210} \\ 0 & 0 & \frac{1}{5}\sqrt{35} \end{bmatrix}$$

Überprüfung der Zerlegung:

```
> A = evalm(q &* R);
```

$$A = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 4 & 5 \\ 2 & 3 & 1 \end{bmatrix}$$

2. Cholesky-Zerlegung

Jede positiv definite Matrix $A$ kann in ein Produkt $A = L\,L^T$ zerlegt werden, wobei $L$ eine untere Dreiecksmatrix mit positiven Diagonalelementen ist.

```
> A := matrix(3,3,[[7,4,3],[4,5,2],[3,2,10]]);
```

$$A := \begin{bmatrix} 7 & 4 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 10 \end{bmatrix}$$

Mit der Funktion **definite** läßt sich überprüfen, ob die Matrix $A$ positiv definit ist:

```
> definite(A, 'positive_def');
```

$$true$$

Zur Berechnung der unteren Dreiecksmatrix $L$ verwenden wir die Funktion **cholesky** :

```
> L := cholesky(A);
```

$$L := \begin{bmatrix} \sqrt{7} & 0 & 0 \\ \frac{4}{7}\sqrt{7} & \frac{1}{7}\sqrt{133} & 0 \\ \frac{3}{7}\sqrt{7} & \frac{2}{133}\sqrt{133} & \frac{1}{19}\sqrt{3135} \end{bmatrix}$$

Überprüfung, ob $A = LL^T$ erfüllt ist:

```
> evalm(L &* transpose(L));
```

$$\begin{bmatrix} 7 & 4 & 3 \\ 4 & 5 & 2 \\ 3 & 2 & 10 \end{bmatrix}$$

3. Singulärwertzerlegung

Die Singulärwertzerlegung einer $m \times n$ Matrix $A$ mit $n \leq m$ hat die Form $A = U\,S\,V^T$ und kann mit der Funktion **Svd** berechnet werden. $U$ ist eine $m \times m$ Orthogonalmatrix, V eine $n \times n$ Orthogonalmatrix, $S$ ist eine $n \times n$ Diagonalmatrix mit den Singulärwerten $0 < s_i$ als Elemente. Die Singulärwerte werden von der Funktion Svd zurückgeliefert. Diese Zerlegung ist in der numerischen Mathematik zur Lösung von Fehlergleichungen von Bedeutung.

```
> A := matrix(3,2,[[1,4],[2,5],[3,6]]);
```

$$A := \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}$$

```
> evalf(Svd(A,'U','V'));
```

$$[9.508032001,\ .7728696357]$$

Überprüfung der Orthogonalität von $U$ und $V$ :

```
> evalm(U);
> evalm(transpose(U) &* U);
```

$$\begin{bmatrix} -.4286671335 & .8059639086 & .4082482905 \\ -.5663069188 & .1123824141 & -.8164965809 \\ -.7039467041 & -.5811990804 & .4082482905 \end{bmatrix}$$

$$\begin{bmatrix} .9999999998 & 0 & -.1\,10^{-9} \\ 0 & 1.000000000 & 0 \\ -.1\,10^{-9} & 0 & 1.000000000 \end{bmatrix}$$

```
> evalm(V);
> evalm(transpose(V) &* V);
```

$$\begin{bmatrix} -.3863177031 & -.9223657801 \\ -.9223657801 & .3863177031 \end{bmatrix}$$

$$\begin{bmatrix} 1.000000000 & 0 \\ 0 & 1.000000000 \end{bmatrix}$$

$U^T\,A\,V$ liefert eine Diagonalmatrix mit den Singulärwerten $s_i$ als Diagonalwerten. Durch Rundungsfehler sind aber die Nebendiagonalen nicht exakt Null.

```
> S := evalm(transpose(U) &* A &* V);
```

$$S := \begin{bmatrix} 9.508032000 & .1\,10^{-8} \\ -.6\,10^{-9} & .7728696355 \\ 0 & 0 \end{bmatrix}$$

$U\,S\,V^T$ müßte bis auf Rundungsfehler wieder die Matrix A ergeben:

```
> evalm(U &* S &* transpose(V));
```

$$\begin{bmatrix} 1.000000000 & 3.999999999 \\ 2.000000000 & 4.999999999 \\ 3.000000000 & 6.000000000 \end{bmatrix}$$

Die Singulärwerte der Matrix $A$ berechnen sich als Quadratwurzel aus den Eigenwerten der Matrix $A^T\,A$ . Für die Singulärwerte erhalten wir somit die exakten Ausdrücke:

```
> map(sqrt, [eigenvals(evalm(transpose(A) &* A))]);
```

$$[\frac{1}{2}\sqrt{182+2\sqrt{8065}},\ \frac{1}{2}\sqrt{182-2\sqrt{8065}}]$$

```
> evalf(%);
```

$$[9.508032000,\ .7728696365]$$

Diese exakten Singulärwerte bekommt man auch direkt mit der Funktion **singularvals**.

# Literaturverzeichnis


[1] K.M. Heal, M. Hansen, K. Rickard, *Maple V Learning Guide*, Springer-Verlag, 1998, ISBN 0-387-98397-X

[2] M.B. Monagan, K.O. Geddes, K.M. Heal, G. Labahn, S.M. Vorkoetter, *Maple V Programming Guide*, Springer-Verlag, 1998, ISBN 0-387-98398-8

[3] A. Heck, *Indroduction to Maple*, 2/e, 1996, Springer-Verlag, ISBN 0-387-94535-0

[4] M. Kofler, *Maple V Release 3: Einführung und Leitfaden für den Praktiker*, 1994, Addison-Wesley (Deutschland) GmbH, ISBN 3-89319-765-6

[5] M. Hörhager, *Maple in Technik und Wissenschaft*, 1996, Addison-Wesley (Deutschland) GmbH, ISBN 3-89319-929-2

[6] T. Westermann, *Mathematik für Ingenieure mit Maple, Band 1: Differential- und Integralrechnung für Funktionen einer Variablen, Vektor- und Matrizenrechnung, Komplexe Zahlen, Funktionenreihen*, 1996, Springer-Verlag, ISBN 3-540-61249-1

[7] W. Werner, *Mathematik lernen mit Maple V: Ein Lehr- und Arbeitsbuch für das Grundstudium*, 1993, ELBI-Verlag, ISBN 3-929694-03-4

# Index